Universidade de Aveiro Departamento de Matemática
2010

JORGE MENDES TAVARES

ANÁLISE E SIMULAÇÃO DE REDES DE COMUNICAÇÕES COM COMUTAÇÃO DE PACOTES EM MATLAB

Universidade de Aveiro Departamento de Matemática
2010

JORGE MENDES TAVARES

ANÁLISE E SIMULAÇÃO DE REDES DE COMUNICAÇÕES COM COMUTAÇÃO DE PACOTES EM MATLAB

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Matemática Aplicada à Engenharia, realizada sob a orientação científica do Doutor Rui Jorge Tomaz Valadas, Professor Catedrático do Departamento de Engenharia Electrotécnica e de Computadores do Instituto Superior Técnico.

Apoio financeiro do Instituto Português
de Apoio ao Desenvolvimento(IPAD)

COOPERAÇÃO PORTUGUESA

Dedico este trabalho ao meu filho William Jorge F.Tavares

O júri

Presidente	Prof. Doutor Domingos Moreira Cardoso
	Prof. Doutor Rui Jorge Tomaz Valadas
	Prof. Doutor António Manuel Nogueira

Agradecimentos

Ao meu orientador, Professor Doutor Rui Jorge Tomaz Valadas, cujos conhecimentos, compreensão e paciência, contribuíram de forma decisiva para a conclusão com sucesso deste trabalho de Mestrado.

À minha família, pelo apoio incondicional que me proporcionou ao longo de toda a minha vida académica o que permitiu-me atingir os meus objectivos académicos e profissionais.

A todos os colegas e amigos que me apoiaram, com os seus conhecimentos.

palavras-chave

Rede, pacote, comunicações, comutação, fila, análise, encaminhamento, modelo, escalonador e simulador.

resumo

A análise e simulação de redes de comunicações permitem levar a cabo estudos de avaliação de desempenho que visam tornar a utilização dos recursos disponíveis mais eficientes. Esta dissertação apresenta métodos de análise e simulação de redes com auxílio do MATLAB, considerando diferentes mecanismos de redes. O documento é composto por três partes. Na primeira parte é feita uma apresentação dos resultados da teoria das filas de espera aplicáveis à modelação de ligações ponto-a-ponto. São também descritos simuladores de eventos discretos para estas ligações e comparados os resultados teóricos com os de simulação. Na segunda parte é apresentada a aproximação de Kleinrock e sua aplicação na modelação de redes com encaminhamento fixo. É também descrito um simulador para este tipo de redes, e são comparados os resultados obtidos com a aproximação e com o simulador. Finalmente, na terceira parte, é apresentado um estudo comparativo de algoritmos de escalonamento que permitem diferenciação da qualidade de serviço em redes, bem como simuladores que incluem alguns destes algoritmos.

keywords Network, package, communications, commutation, line, analysis, guiding, model, escalonador and simulator

abstract The analysis and the simulation of the communication networks allow studies of performance evaluation with the main objective of making the use of available resources more efficient. This thesis presents methods of analysis and simulation of networks with the aid of MATLAB taking into account different mechanism networks. The document comprises three parts. The first part is concerned with the presentation of the results of the theory of waiting lines position for the modeling of point-to-point. The discrete event simulators are also described for these links and compare the simulation with the theoretical results. The second part presents the Kleinrock´s approach and its application in modeling networks with fixed routing. It is also described as a simulator for such networks, and the results are compared with the approach and the simulator. Finally, the third part presents a comparative study of scheduling algorithms that allow differentiation of quality of network services as well as simulators that include some of these algorithms.

ÍNDICE GERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

CAPÍTULO I - INTRODUÇÃO

Introdução

A comunicação, de uma forma geral, vem constituindo um instrumento amplamente utilizado na integração e desenvolvimento da sociedade. A investigação realizada em torno das redes de comunicações de dados após o surgimento do tráfego telefónico tem sofrido rápidos avanços impulsionando o desenvolvimento da Engenharia de Tráfego.
Possibilitam o aperfeiçoamento de estudos analíticos e computacionais sobre as mais diversas formas de processar dados e de os transmitir com o objectivo de melhorar os sistemas de redes, permitindo, portanto, solucionar ou minimizar problemas tais como: atrasos, perdas de dados, entre outros aspectos perturbadores do normal funcionamento das redes, que naturalmente podem ocorrer nos diferentes elementos que constituem um sistema de comunicação de dados.
A Engenharia de Tráfego permite ainda, fazer simulações de factos que posteriormente serão materializados, reforçando deste modo, a ideia da rentabilização dos recursos disponíveis no âmbito do sistema de transmissão de dados.

1.1 Contexto e Motivações

Este estudo tem a sua génese no intersse particular do investigador em alargar os conhecimentos nas áreas de redes de comunicações com comutação de pacotes, interesse esse reavivado com o desenrolar da componente curricular do curso de Mestrado em Matemática Aplicada à Engenharia.
A investigação realizada em torno das redes de comunicações de dados tem conduzido a grandes avanços, nomeadamente ao aparecimento e consolidação da Internet. Para tornar a utilização das redes de comunicações mais eficiente, é necessário estabelecer modelos matemáticos que as descrevam ou recorrer a simulação em computador. Esta

dissertação propõe-se abordar esta área do conhecimento, onde se cruzam a Matemática e a Engenharia.

1.2 Finalidades e Questões de Investigação

Esta investigação tem por finalidade, efectuar um estudo qualitativo e quantitativo das características das redes de comunicações com comutação de pacotes.
Para o efeito, tornam-se necessário definir alguns conceitos fundamentais, tais como:

- Redes de comunicações;
- Comutação de pacotes;
- Encaminhamento;
- Diferenciação da qualidade de serviço.

Pretende-se, ao longo desta dissertação dar respostas a esses conceitos e outros considerados pertinentes.

1.3 Estrutura da Dissertação

Este trabalho é composto por seis capítulos. Em cada um, procura-se, sempre que possível, dar exemplos analíticos e posteriormente implementados em MATLAB.
No segundo capítulo, é apresentado um breve historial a cerca do surgimento de redes com comutação de pacotes, bem como os avanços conseguidos neste domínio principalmente no que diz respeito às topologias que foram tomando estruturas diferentes até chegar à estrutura distribuída que conferem maior robustez ao sistema (KLEINROCK, 1961). Abordam-se ainda, questões importantes na análise de redes de comunicações, como são as teorias de filas de espera e outros conceitos relacionados, que servem de base para o entendimento e a análise de ligações ponto-a-ponto e os vários mecanismos de encaminhamento, onde alguns modelos, como M/M/1, M/D/1 e M/G/1 são analisados e exemplificados analiticamente. No terceiro capítulo, descrevem-se as rotinas, as estruturas de dados e o fluxo de processamento relativos aos mecanismos ou programas de simulação usando o MATLAB, que servem de suporte

para a confirmação dos resultados analíticos encontrados no capítulo anterior. No quarto capítulo, faz-se referência ao encaminhamento fixo, onde é apresentado o Teorema da Independência de Leonard Kleinrock, bem como a sua aplicação no processo de encaminhamento de pacotes. O capítulo cinco foi reservado à implementação em MATLAB de diversas rotinas cuja principal finalidade é confirmar os resultados analíticos encontrados ao longo deste trabalho. Aqui são ainda descritas as rotinas, as estruturas de dados e o fluxo de processamento. Mostra-se as limitações da aproximação de Kleinrock numa ligação ponto-a-ponto quando a tráfego é intenso. Alguns algoritmos usados nas redes com diferenciação de qualidade de serviço, como por exemplo, o algoritmo de escalonamento FIFO, o algoritmo de prioritização estrita e o deficit-round-robin, são referenciados no sexto capítulo, bem como os modos de funcionamento e as vantagens e desvantagens dos mesmos.

CAPÍTULO II – ANÁLISE DE LIGAÇÕES PONTO-A-PONTO

Introdução

A avaliação de desempenho das redes de comunicações tem por base a teoria dos processos estocásticos. A comutação de pacotes é uma alternativa à comutação de circuitos. A Internet é baseada em comutação de pacotes. Neste tipo de redes de comunicações o desenvolvimento de modelos para análise de desempenho requer o conhecimento da estrutura da rede e do tráfego oferecido.

Propõe-se, neste capítulo formular as equações matemáticas para os diferentes sistemas de encaminhamento de dados e apresentar exemplos que posteriormente serão simulados.

2.1 SURGIMENTO DAS REDES DE COMUNICAÇÕES COM COMUTAÇÃO DE PACOTES

As redes de comunicações com comutação de pacotes foram estabelecidas nos anos 60 [fig. 2.1]. Comutação de pacotes é um método de fragmentar a mensagem em partes [fig. 2.2], denominados pacotes e encaminhados depois ao seu destino [fig. 2.4], voltando a reagrupá-los [fia 2.5 e 2.6]. O método de empacotamento da mensagem facilita a partilha da mesma ligação por vários utilizadores, fraccionando a mensagem em unidades discretas, que podem ser reencaminhadas separadamente. Nesta perspectiva, esta tecnologia permite que um pacote que não tenha chegado ao seu destino possa ser reencaminhado enquanto outros bons são transmitidos ininterruptamente. (HARDY, (s/d)).

Os pacotes podem levar informação sobre eles mesmos [fig.2.3], donde vêem e para onde vão, podem ser comprimidos para melhorar a velocidade ou podem ser cifrados

por motivos de segurança. A maior parte dos pacotes leva consigo algum tipo de verificação de consistência que ajuda a detectar pacotes com erros. O desenvolvimento de redes de comutação de pacotes teve alguns precedentes nos primeiros sistemas de tempo partilhado operados por companhias como a [1]IBM. Estes sistemas permitiam a ligação remota aos seus computadores durante um determinado período de tempo. Os sistemas de tempo partilhado geralmente ofereciam uma relação cliente/servidor, enquanto as redes com comutação de pacotes, embora estruturadas hierarquicamente, são essencialmente redes ponto-a-ponto, pelo que representou uma revolução na maneira de ver os computadores e veio a pôr em causa a eficiência, a eficácia dos grandes sistemas centralizados de partilha de tempo [fig. 2.7], em favor de sistemas pequenos todos ligados entre si, partilhando recursos existentes.

Os grandes sistemas centralizados ainda são usados em tarefas específicas. O estabelecimento das redes de comutação de pacotes ponto-a-ponto contribui para a democratização do uso da computação.

Em Julho de 1961, Leonard Kleinrock publicou o primeiro artigo (KLEINROCK, 1961) acerca da teoria de redes com comutação de pacotes e em 1964, o 1º livro sobre o assunto (KLEINROCK, 1964). Devido ao aspecto inovador deste trabalho, Leonard Kleinrock é actualmente considerado como o "Pai da Internet", e 1969, o ano em que o seu desenvolvimento teórico deu lugar à implementação da [2]ARPANET, é considerado o ano do "nascimento da Internet".

> [3]"29 de Outubro de 1969, 22h30. A primeira ligação da ARPANET, rede que deu origem à Internet, era estabelecida entre a Universidade da Califónia (UCLA), em Los Angeles e o Stanford Research Institute (SRI), no mesmo estado norte-americano. O objectivo era enviar um pacote de dados de um computador para outro, que consistia apenas numa expressão "LOG IN". À medida que na UCLA iam escrevendo letra a letra, do SRI confirmavam por telefone a recepção. Chegou o "L", chegou o "O", mas o "G" ficou pelo caminho. O sistema bloqueou e a experiência ficou por aí.
>
> Apesar da interrupção, *"fiquei emocionado por termos conseguido esta mensagem sucinta e profética. Afinal, dizia: "Olhai!" [tradução de Lo!] como que "Olhai e vede!" ["Lo and behold!"]. Não se podia esperar uma*

[1] IBM – International Business Magchines.
[2] Advanced Research Projects Agency Network.
[3] Parte do documento publicado a 29-10-2009, 13:15 pela SIC on line. Disponível em: http://www.dragteam.info/forum/noticias-informacoes/95578-internet-nasceu-ha-40-anos.html. Consultado em 15 de Fevereiro de 2010.

mensagem mais sucinta e profética, e foi acidental", disse à SIC o responsável pela experiência, Leonard Kleinrock."

Paul Baran no seu trabalho publicado em 1964 (BARAN, 1964) sugeriu, que as redes com comutação de pacotes deveriam ser desenhadas como redes distribuídas, ao contrário de redes centralizadas e descentralizadas que eram os modelos básicos de construção de redes de comunicações na altura [Figura 2.7]. Esta nova forma de representar redes, fez com que tornassem mais robustas, consistentes e por conseguinte muito mais confiáveis principalmente porque, em caso de avaria em um ou alguns dos nodos que compõem a rede, o sistema pode ainda ter um comportamento aceitável, o que não é possível com os modelos anteriormente mencionados. Segundo Baran, a rede distribuída não devia possuir um sistema central. A distribuição dos nodos seria tal que, cada um devia estar ligado a vários outros, numa espécie de entrelaçado. Assim, cada nó tinha vários caminhos para enviar as mensagens. Em caso de defeito em um dos caminhos, outro poderia ser usado para transmitir mensagens.

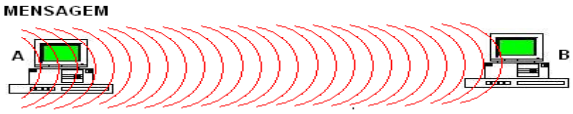


Figura 2.1 - Mensagens serão enviadas do computador A para B através da rede.

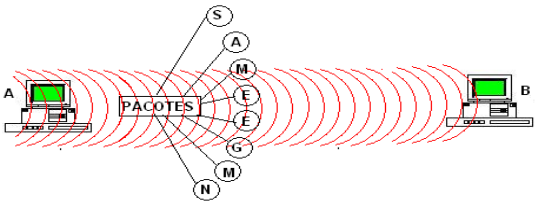


Figura 2.2 - A mensagem é dividida em pacotes.

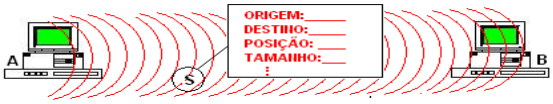


Figura 2.3 – Os pacotes são rotulados individualmente.

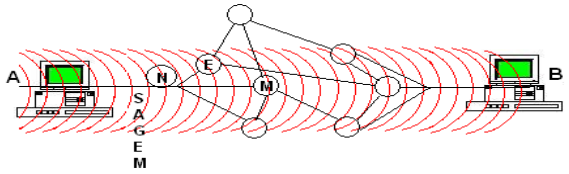


Figura 2.4 – Pacotes enviados sequencialmente pela rede.

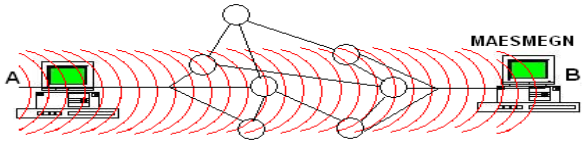


Figura 2.5 – Os pacotes passam por vários nodos até chegarem ao destino.

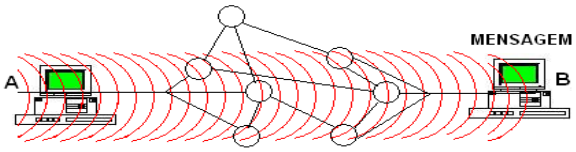


Figura 2.6 – O computador B reordenar a mensagem

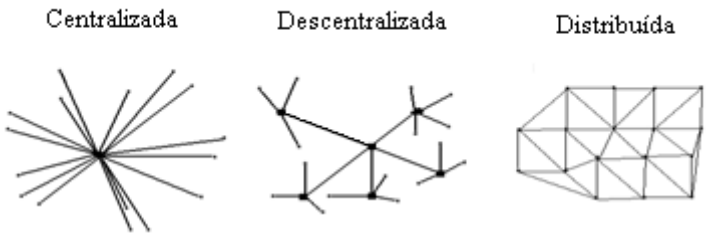


Figura 2.7 – Topologias de rede

2.2 TEORIAS DE FILAS NA MODELAÇÃO DE REDES DE COMUNICAÇÕES

Na execução de redes de comunicações [fig. 2.9], há sempre a preocupação com o desempenho dos processadores, com a capacidade de armazenar e transferir pacotes. Existem vários métodos para fazer o estudo do comportamento de uma rede de comunicação de dados, por exemplo, através do estudo analítico ou por simulação. A teoria de filas de espera permite a definição de equações com aplicações directas na análise do desempenho de redes, pois a utilização apropriada dessas equações permite encontrar estimativas dos parâmetros de qualidade de serviço mais importantes, motivo pelo qual o estudo analítico das redes é normalmente baseado em modelos desenvolvidos a partir da teoria de filas de espera (CERF, 1974). Torna-se necessário e oportuno fazer referência aos conceitos básicos e essenciais desta teoria mais utilizados em redes de transmissão de dados, bem como a aplicação desses modelos no estudo de redes de comunicações.

O processo de Poisson fundamentado em incrementos independentes é amplamente utilizado na modelação estocástica de redes de comunicações de dados. A modelação das redes de comunicações tem fundamentos na teoria das filas de espera. Criar, analisar e interpretar a disciplina das filas são aspectos necessários e indispensáveis neste contexto, bem como a definição dos mecanismos de encaminhamento e atendimento dos pacotes. A rede é normalmente usada por vários fluxos, em que cada fluxo segue um único caminho, que por sua vez, consiste em uma sequência de ligações entre a origem e o destino. Esta forma de transmitir dados está sujeita a algumas simplificações conforme sugere Leonard Kleinrock, assunto que será abordado no quarto capítulo.

Formam-se filas de espera [fig. 2.8] no processo de transmissão de dados, sempre que o número de clientes é maior do que o número de servidores para um determinado instante e a média do processo de chegada seja maior do que a média do processo de atendimento, isto é, o servidor demora um certo tempo para atender cada cliente (tempo de serviço). Em muitas situações os clientes encontram o servidor ocupado e têm que aguardar algum tempo numa fila de espera.

A aplicação da teoria de filas de espera ultrapassa o campo das comunicações e tem aplicações nas mais distintas áreas. No entanto, a estrutura básica e genérica de uma fila de espera é composta pelos seguintes elementos:

- A fonte de entrada, que tem como função modelar o processo de chegada dos clientes;
- A fila de espera, que por sua vez modela o lugar onde os clientes aguardam pelo serviço.
- A disciplina da fila que estabelece o critério para escolher a ordem que os clientes da fila serão servidos.
- O atendimento (Serviço) tem como função modelar o sistema de atendimento dos clientes.

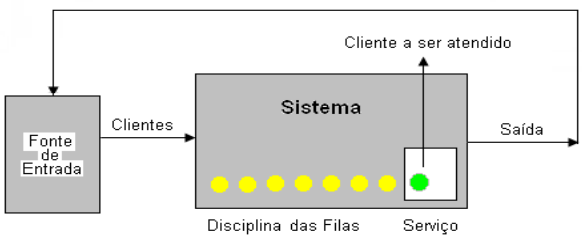
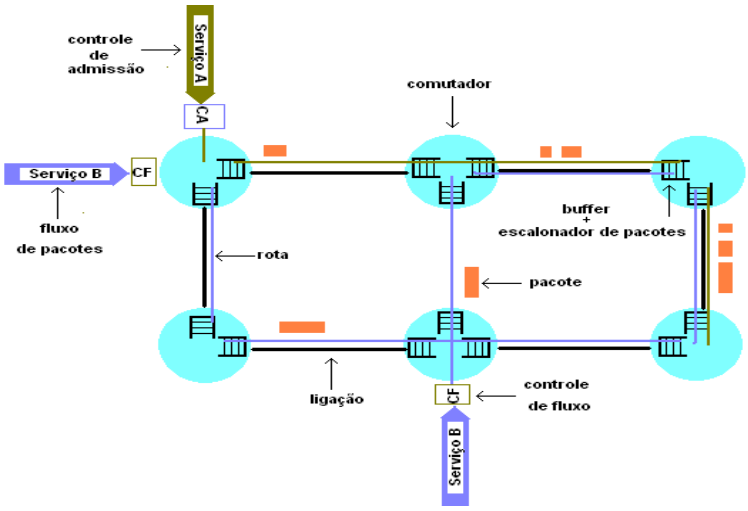


Figura 2. 8 - Estrutura básica de um modelo de fila



[4]Figura 2.9 – Redes de comunicações - Funções básicas

[4] Buffer – meio de armazenamento ou rotina usada para compensar a diferença na taxa de fluxo de dados entre dispositivos
Switch (comutador) – dispositivo utilizado em redes de computadores para reencaminhar módulos (frames) entre os diversos nós.

2.2.1 Processos de nascimento e morte

Um processo de nascimento e morte é um caso particular de um processo de Markov. Por sua vez, um processo de Markov, é um processo estocástico em que o estado futuro do processo dado o presente é independente do passado (SHELDON, 1997).
Os processos de nascimento e morte estão na base dos sistemas de filas de espera mais simples.

Considerando que o estado do sistema é o número de clientes (n) no sistema e Δt é o intervalo de observação verifica-se que:

1. Para $n > 0$ e $\Delta t \rightarrow 0$, existem apenas três possibilidades para o estado seguinte:
   - Passar para o estado $n+1$ se um cliente chegar ao sistema no intervalo de tempo Δt a uma taxa de chegada λ_n.
   - Passar para o estado $n-1$ se um cliente foi atendido pelo sistema no intervalo Δt a uma taxa de atendimento μ_n.
   - Permanecer no estado n se nenhumas das duas condições anteriores forem observadas no intervalo Δt.
2. Para $n = 0$ e $\Delta t \rightarrow 0$, existem apenas duas possibilidades para o próximo estado do sistema:
   - Estado $n = 1$ se um cliente chegar ao sistema no intervalo de tempo Δt a uma taxa de chegada λ_0.
   - Continua no estado $n = 0$ se neste intervalo de tempo nenhum cliente chegar ao sistema.

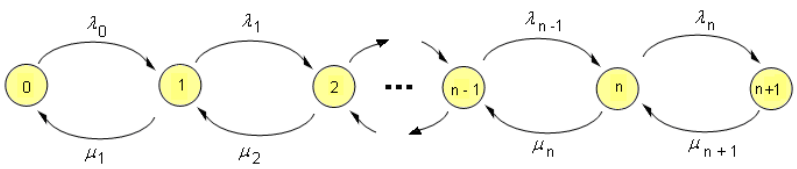


Figura 2.10 – Diagrama de transição de estado de um processo de nascimento e morte

Considerando que num sistema estável [fig. 2.10] a taxa esperada de fluxo que entra no estado *n* é igual à que sai do mesmo estado, então para o estado *n = 0*, tem-se a seguinte equação de balanço do processo de nascimento e morte:

$$\mu_1\pi_1 = \lambda_0\pi_0$$

$$\Rightarrow \pi_1 = \frac{\lambda_0}{\mu_1}\pi_0 \qquad 2.1$$

Procedendo da mesma forma para o estado n = 1, tem-se:

$$\lambda_0\pi_0 + \mu_2\pi_2 = (\mu_1 + \lambda_1)\pi_1$$

$$\Rightarrow \pi_2 = \frac{\lambda_1\lambda_0}{\mu_1\mu_2}\pi_0 \qquad 2.2$$

Pelo princípio da indução, pode-se mostrar que:

$$\pi_n = \frac{(\lambda_{n-1}\lambda_{n-2}...\lambda_1\lambda_0)}{(\mu_n\mu_{n-1}...\mu_2\mu_1)}\pi_0 \qquad 2.3$$

A probabilidade de uma cadeia de Markov em tempo contínuo estar no estado *n* após um intervalo de tempo *t* converge frequentemente para um valor limite independente do estado inicial.

As probabilidades limite são:

Como $\sum_{n=0}^{\infty}\pi_n = 1$ então

$$\pi_0 + \sum_{n=1}^{\infty}\pi_n = 1 \Rightarrow \pi_0 + \sum_{n=1}^{\infty}\frac{(\lambda_{n-1}\lambda_{n-2}...\lambda_1\lambda_0)}{(\mu_n\mu_{n-1}...\mu_2\mu_1)}\pi_0 = 1$$

$$\Rightarrow \pi_0 = \frac{1}{1+\sum_{n=1}^{\infty}\frac{(\lambda_{n-1}\lambda_{n-2}...\lambda_1\lambda_0)}{(\mu_n\mu_{n-1}...\mu_2\mu_1)}} \qquad 2.4$$

Pelo princípio da indução, pode-se mostrar que:

$$\pi_n = \frac{\lambda_{n-1}\lambda_{n-2}...\lambda_1\lambda_0}{\mu_n...\mu_2\mu_1\left(1+\sum_{n=1}^{\infty}\frac{(\lambda_{n-1}\lambda_{n-2}...\lambda_1\lambda_0)}{(\mu_n\mu_{n-1}...\mu_2\mu_1)}\right)}, n \geq 1$$

$$\Rightarrow \pi_n = \frac{\frac{\lambda_{n-1}\lambda_{n-2}...\lambda_1\lambda_0}{\mu_n...\mu_2\mu_1}}{\left(1+\sum_{n=1}^{\infty}\frac{(\lambda_{n-1}\lambda_{n-2}...\lambda_1\lambda_0)}{(\mu_n\mu_{n-1}...\mu_2\mu_1)}\right)}, n \geq 1 \qquad 2.5$$

Um processo de Poisson é em caso particular de um processo de nascimento e morte em que apenas existem nascimentos.

2.2.2 Distribuição de Poisson

Esta função foi introduzida por Poisson como forma limite de estudar as propriedades binomiais.

A probabilidade de um numero designado de sucessos por unidade de intervalo, P (X), pode ser encontradapor:

$$P(X) = \frac{\lambda^X e^{-\lambda}}{X!} \qquad 2.6$$

onde,

X: o número designado de sucessos;

λ : o número médio de sucessos num intervalo específico;

e : a base do logaritmo natural.

2.2.3 Teorema de Little

Para demonstrar este Teorema algumas definições são necessárias.

Admita que um sistema foi observado do tempo t = 0 até um tempo indefinido, e que os valores das variáveis de interesse tenham sido registados, em particular as variáveis:

- $N(t)$ - Número de clientes que chegaram no intervalo [0,t].
- $L(t)$ - Número de clientes no sistema no instante t.
- W_i - Tempo médio de espera na fila do iésimo cliente.

O número médio de clientes no sistema N, a taxa média de chegada dos clientes no sistema λ e o tempo médio de atraso dos clientes no sistema W, podem ser encontrados pelas seguintes equações respectivamente, assumindo que tais limites existam:

A média temporal do número de clientes observados até ao instante t é:

$$L_t = \lim_{t\to\infty} \frac{1}{t} \int_0^t L(\tau) d\tau \qquad 2.7$$

L_t, varia com t, mas em muitos sistemas de interesse, tende para um valor estacionário. A média temporal em estado estacionário (ou simplesmente média temporal) é:

$$L = \lim_{t\to\infty} L_t \qquad 2.8$$

A média temporal da taxa de chegada no intervalo [0,t] é $\lambda_t = N(t)/t$ e a média temporal em estado estacionário é portanto:

$$\lambda = \lim_{t\to\infty} \frac{N(t)}{t} \qquad 2.9$$

A média temporal do atraso dos clientes até ao instante t é:

$$W_t = \frac{\sum_{i=0}^{N(t)} W_i}{N(t)} \qquad 2.10$$

e a média temporal em atraso estacionário $W = \lim_{t\to\infty} W_t$ 2.11

As quantidades acima relacionam-se através de uma fórmula muito simples conhecida como teorema de Little, que diz que o número médio de clientes no sistema L é igual ao produto da taxa média de chegada ao sistema λ pelo atraso médio de espera no sistema W:

$$L = \lambda W \qquad 2.12$$

O Teorema de Little traduz a ideia intuitiva de que sistemas congestionados (L elevado) estão associados a grandes atrasos (W elevado). Num dia de chuva, o tráfego em hora de ponta é mais lento do que normalmente (W elevado) e as ruas estão mais congestionadas (L elevado). Um restaurante de refeições rápidas (W pequeno) precisa

de uma sala menor (L pequeno) do que um restaurante normal, para a mesma taxa de chegada de clientes (mesmo λ).

2.3 ANÁLISE DE LIGAÇÕES PONTO-A-PONTO ATRAVÉS DOS MODELOS M/M/1, M/D/1 E M/G/1 COM E SEM PRIORIDADES

2.3.1 Notação de Kendall

Um sistema de fila de espera é descrito por uma série de símbolos separados por barra. A representação tem a forma A/B/X/Y/Z, em que, A indica a distribuição do intervalo entre chegadas, B a distribuição do tempo de serviço, X o número de servidores em paralelo, Y a capacidade do sistema e Z a disciplina de serviço. Este tipo de notação deve-se essencialmente a Kendall (KESHAV e WESLEY, 1997).
O símbolo M indica a distribuição exponencial (a utilização de M advém da propriedade Markoviana da distribuição exponencial); G indica uma distribuição Geral, que represente variáveis aleatórias independentes e identicamente distribuídas; D indica uma característica determinística. Em muitas situações apenas os primeiros três símbolos são utilizados. A prática usual é omitir Y em sistemas de capacidade infinita (Y = ∞) e omitir Z quando a disciplina de serviço é do tipo FIFO (First-In-First-Out).

2.3.2 O sistema M/M/1

Neste sistema o processo de chegada é de Poisson, o tempo de serviço tem distribuição exponencial, existe um único servidor, a capacidade da fila é infinita, o intervalo entre chegadas de clientes ao sistema e o tempo de serviço são estatisticamente independentes. Para este sistema considera-se que a taxa de chegada e a taxa de serviço são independentes do estado em que se encontra o sistema, ou seja:

$$\lambda_n = \lambda, \forall n$$
$$\mu_n = \mu, \forall n$$

Sendo assim, a equação 2.5 reduz-se a:

$$\pi_n = \frac{\left(\lambda/\mu\right)^n}{\sum_{n=0}^{\infty}\left(\frac{\lambda}{\mu}\right)^n} \qquad 2.13$$

onde $\rho = \frac{\lambda}{\mu}$ é definido como sendo (factor de utilização do sistema).

O numero médio de clientes no sistema é dado por:

$$L = 0.\pi_0 + 1.\pi_1 + 1.\pi_2 + ... = \sum_{n=0}^{\infty} n\pi_n = \sum_{n=0}^{\infty} n\left(\frac{\lambda}{\mu}\right)^n\left(1-\frac{\lambda}{\mu}\right) =$$

$$\left(1-\frac{\lambda}{\mu}\right)\sum_{n=0}^{\infty} n\left(\frac{\lambda}{\mu}\right)^n = \left(1-\frac{\lambda}{\mu}\right)\frac{\frac{\lambda}{\mu}}{\left(1-\frac{\lambda}{\mu}\right)^2}$$

$$\Rightarrow L = \left(\frac{\lambda}{\mu-\lambda}\right) \qquad 2.14$$

Utilizando o teorema de Little e a equação 2.14, podemos calcular o tempo médio que cada cliente aguarda no sistema pela equação:

$$W = \frac{L}{\lambda} = \frac{1}{\mu-\lambda} \qquad 2.15$$

Atraso médio na fila de espera

$$W_Q = W - E[S] = W - \frac{1}{\mu} = \frac{\lambda}{\mu(\mu-\lambda)} \qquad 2.16$$

Número médio de clientes na fila de espera:

$$L_Q = \lambda W_Q = \frac{\lambda^2}{\mu(\mu-\lambda)} \qquad 2.17$$

Factor de utilização do servidor:

$$\rho = \frac{\lambda}{\mu} \qquad 2.18$$

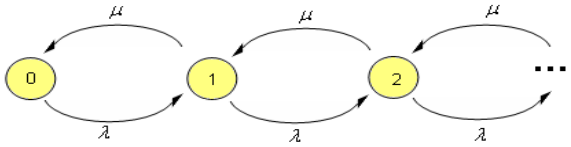


Figura 2.11 – Diagrama de transição de estados, M/M/1

2.3.2.1 Exemplo de sistema M/M/1

Como exemplo da utilização do modelo M/M/1, considera-se uma ligação ponto-a-ponto entre dois terminais A e B (fig. 2.12). O terminal A (Fonte) transmite e o B (Receptor) recebe dado, por uma ligação de 10 Kbps. O processo de chegada de pacotes (clientes) ao sistema é um processo de Poisson, o buffer é infinito e o tamanho dos pacotes tem uma distribuição exponencial. A capacidade da ligação é constante, portanto, o tempo de serviço de cada pacote depende do tamanho do pacote, ou seja, o tempo de serviço dos pacotes é uma variável aleatória com distribuição exponencial. O tamanho médio dos pacotes é 500 bits. A taxa de chegada em pacotes por segundo é igual a 16, o que corresponde ao valor de λ. A taxa de serviço é igual a capacidade da ligação dividido pelo tamanho médio dos pacotes, ou seja, 20 pacotes por segundo que corresponde ao valor de μ.

Pretende-se determinar:

- O número médio de pacotes no sistema?
- O atraso médio no sistema?
- O atraso médio na fila de espera?
- O número médio de pacotes na fila de espera?
- O factor de utilização do servidor?

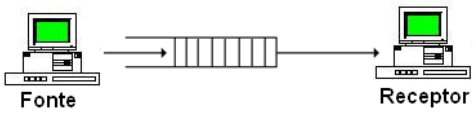


Figura 2.12 – Exemplo de sistema M/M/1

Tabela 2.1– Resultados do exemplo do sistema M/M/1

	Valores Calculados	Valores Simulados (8 simulações)										
		1	2	3	4	5	6	7	8	Média	Variância	Intervalo de confiança a 90%
O número médio de pacotes no sistema é dado: • Pela equação 2.14	4 Pacotes	3,9478	4,4508	3,8377	3,775	4,1729	4,2452	3,5946	3,8913	3,9894	0,0783	[3.8267,4.1522]
O atraso médio no sistema é dado: • Pela equação 2.15	0,25*s*	0,2493	0,2774	0,2401	0,2349	0,2601	0,2636	0,2249	0,2439	0,2493	0,0003	[0.2394,0.2592]
O atraso médio na fila de espera é dado: • Pela equação 2.16	0,2*s*	0,1993	0,2274	0,1901	0,1849	0,2101	0,2136	0,1749	0,1939	0,1993	0,0003	[0.1894,0.2092]
O número médio de pacotes na fila de espera é dado: • Pela equação 2.17	3,2 Pacotes	3,1603	3,6480	3,0455	2,977	3,3594	3,4399	2,8009	3,0870	3,1898	0,0757	[3.0297,3.3498]
O factor de utilização do servidor é dado: • Pela equação 2.18	0,8	0,7876	0,8028	0,7923	0,798	0,8135	0,8053	0,7937	0,8043	0,7997	0,0001	[0.7948,0.8046]

2.3.3 O sistema M/G/1 e M/D/1

No sistema M/G/1, assume-se que o processo de chegada é de Poisson, que existe um único servidor, a fila tem capacidade infinita, o tempo de serviço tem uma distribuição genérica e os clientes são servidos na mesma ordem que chegam à fila (KESHAV e WESLEY, 1997).

Defina-se trabalho num sistema em qualquer instante t, como a soma dos tempos de serviço remanescentes de todos os clientes que estão no sistema no instante t. Seja *V* a média temporal do trabalho no sistema. Recordemos a generalização do teorema de Little:

taxa média à qual o sistema ganha = $\lambda \times$ quantia média paga por cada cliente

Considere-se a seguinte regra de custo: cada cliente paga a uma taxa de *y* por unidade de tempo, quando o seu tempo remanescente de serviço for *y*. Neste caso,

$$V = \lambda E \text{ [quantia paga por cliente]} \qquad 2.19$$

Sejam S e W_Q^* o tempo de serviço e o tempo que um cliente permanece na fila de espera. Um cliente paga a uma taxa constante S por unidade de tempo enquanto na fila de espera e a uma taxa S - x depois de despender um tempo x em serviço. Assim,

E [quantia paga por cliente] = $E\left[SW_Q^* + \int_0^S (S-x)dx \right]$ 2.20

e portanto,

$$V = \lambda E\left[SW_Q^*\right] + \frac{\lambda E\left[S^2\right]}{2} \qquad 2.21$$

Se o tempo de serviço e o tempo de espera na fila forem independentes

$$V = \lambda E\left[S\right] W_Q + \frac{\lambda E\left[S^2\right]}{2} \qquad 2.22$$

Um sistema M/G/1 assume (i) chegadas de Poisson à taxa λ, (ii) uma distribuição de serviço genérica e (iii) um único servidor. Admite-se também que os clientes são servidos pela ordem de chegada.

Para um cliente arbitrário num sistema M/G/1

tempo de espera na fila de espera =

trabalho no sistema quando o cliente chega

e, tomando a média,

W_Q = trabalho médio visto por uma chegada

Devido às chegadas de Poisson, o trabalho médio visto por uma chegada é igual à média temporal do trabalho no sistema V. Assim, $W_Q = V$.

Daqui resulta a fórmula de Pollaczek – Khintchine:

$$W_Q = \frac{\lambda E\left[S^2\right]}{2(1-\lambda E[S])} \qquad 2.23$$

Número médio de clientes na fila de espera:

$$L_Q = \lambda W_Q = \frac{\lambda^2 E\left[S^2\right]}{2(1-\lambda E[S])} \quad 2.24$$

Atraso médio no sistema:

$$W = W_Q + E[S] = \frac{\lambda E\left[S^2\right]}{2(1-\lambda E[S])} + E[S] \quad 2.25$$

Número médio de clientes no sistema:

$$L = \lambda W = \frac{\lambda^2 E\left[S^2\right]}{2(1-\lambda E[S])} + \lambda E[S] \quad 2.26$$

Quando os tempos de serviço são exponencialmente distribuídos, como no sistema M/M/1, resulta

$E[S^2] = \frac{2}{\mu^2}$. Neste caso 2.27

$$W_Q = \frac{\rho}{\mu(1-\rho)} \quad 2.28$$

Quando os tempos de serviço são idênticos para todos os clientes, todos os pacotes com o mesmo comprimento (sistema M/D/1), a variância do processo é zero, logo:

$E[S^2] = \frac{1}{\mu^2}$. 2.29

Neste caso

$$W_Q = \frac{\rho}{2\mu(1-\rho)} \quad 2.30$$

O atraso médio no sistema é dado por:

$$W = \frac{\lambda}{2\mu(\mu-\lambda)} + \frac{1}{\mu} \quad 2.31$$

O número médio de clientes na fila de espera é dado por:

$$L_Q = \frac{\rho^2}{2(1-\rho)} \quad 2.32$$

O número médio de clientes no sistema é dado por:

$$L = \frac{\rho^2}{2(1-\rho)} + \rho \qquad 2.33$$

Uma ligação ponto-a-ponto com capacidade μ pacotes /*s* onde chegam pacotes a uma taxa de Poisson λ pacotes/*s* é um sistema M/G/1. Se o comprimento dos pacotes for exponencialmente distribuído degenera num sistema M/M/1. Se o comprimento dos pacotes for fixo degenera num sistema M/D/1.

2.3.3.1 Exemplo da utilização do modelo M/D/1

Como exemplo de utilização de modelo M/D/1, pode-se utilizar o mesmo exemplo utilizado para o caso do modelo M/M/1 considerando que para este modelo os pacotes têm todos o mesmo tamanho (500 bits). Como o tamanho dos pacotes e a capacidade da ligação são constantes, o tempo de serviço dos pacotes é determinístico.

Pretende-se determinar:

- O número médio de pacotes no sistema?
- O atraso médio no sistema?
- O atraso médio na fila de espera?
- O número médio de pacotes na fila de espera?
- O factor de utilização do servidor?

Tabela 2.2 - Resultados do exemplo do sistema M/D/1

	Valores Calculados	Valores Simulados (8 simulações)										
		1	2	3	4	5	6	7	8	Média	Variância	Intervalo de confiança para 90%
O número médio de pacotes no sistema é dado: • Pela equação 2.14	2,4 Pacotes	2,5229	2,2890	2,3160	2,4024	2,3955	2,3867	2,6196	2,3452	2,4097	0,0122	[2.3455,2.4738]
O atraso médio no sistema é dado: • Pela equação 2.15	0,15s	0,1567	0,1433	0,1442	0,1497	0,1496	0,1496	0,162	0,1457	0,1501	0,000041	[0.1464,0.1538]
O atraso médio na fila de espera é dado: • Pela equação 2.16	0,1s	0,1064	0,0933	0,0942	0,0997	0,0996	0,0996	0,112	0,0957	0,1001	0,000040	[0.0964,0.1038]
O número médio de pacotes na fila de espera é dado: • Pela equação 2.17	1,6 Pacotes	1,7162	1,4707	1,5134	1,600	1,5947	1,5893	1,8110	1,5407	1,6045	0,0122	[1.5401,1.6689]
O factor de utilização do servidor é dado: • Pela equação 2.18	0,8	0,8067	0,7881	0,8026	0,8024	0,8007	0,7974	0,8086	0,8045	0,7011	0,0803	[0.7977,0.8051]

2.3.4 Sistema M/G/1 com prioridades

Consideremos a possibilidade de servir diferentes fluxos de pacotes de forma diferenciada. Uma possibilidade é atribuir prioridades aos fluxos, de tal forma que os pacotes de um fluxo com maior prioridade sejam sempre servidos antes do que os pacotes de um fluxo com menor prioridade. Este esquema é por vezes designado de prioritização estrita. Note-se que, na eventualidade da fila de espera conter sempre pacotes de um fluxo de maior prioridade, o fluxo de menor prioridade nunca será servido. O sistema M/G/1 com prioridades pode ser utilizado para modelar este tipo de sistema (KESHAV e WESLEY, 1977).

Considere um sistema M/G/1 em que existem n classes de serviço. A k-ésima classe é determinada pela taxa de chegadas, λ_k, pelo primeiro e segundo momentos do tempo de serviço,

$$E(S_k) = \frac{1}{\mu_k} \qquad 2.34$$

e $E(S_k^2)$, e pela prioridade: quanto menor a classe maior a prioridade; a classe 1 tem a prioridade mais elevada. O servidor serve primeiro os clientes das classes com maior prioridade. Os clientes de uma mesma classe são servidos por ordem de chegada (FIFO - First In First Out). Admite-se que as chegadas de cada classe são independentes e de Poisson e independentes dos tempos de serviço. O serviço de um cliente não é interrompido pela chegada de um cliente de uma classe com maior prioridade. Esta disciplina de serviço é designada por não-preemptiva (non-preemptive).

O atraso médio na fila de espera correspondente à classe *k* é dado por:

$$W_{Qk} = \frac{\sum_{i=1}^{n} \lambda_i E(S_i^2)}{2(1-\rho_1-...-\rho_{k-1})(1-\rho_1-...-\rho_k)}, \qquad 2.35$$

onde $\rho_k = \frac{\lambda_k}{\mu_k}$. Assume-se que $\rho_1 + ... + \rho_n < 1$. Se os tempos de serviço de cada classe forem exponencialmente distribuídos com média *1/μ* (sistema M/M/1 do tipo não-preemptivo), resulta

$$W_{Qk} = \frac{\rho / \mu}{(1-\rho_1-...-\rho_{k-1})(1-\rho_1-...-\rho_k)}, \qquad 2.36$$

onde $\rho = \rho_1 + ... + \rho_n$. Se os tempos de serviço forem determinísticos e iguais a 1/μ (sistema M/D/1 do tipo não-preemptivo), resulta:

$$W_{Q_k} = \frac{\rho / 2\mu}{(1-\rho_1-...-\rho_{k-1})(1-\rho_1-...-\rho_k)}, \qquad 2.37$$

Num sistema com disciplina de serviço preemptiva o serviço de um cliente é interrompido quando da chegada de um cliente com maior prioridade; este último entrará em serviço imediatamente; o serviço do primeiro cliente continua do ponto em que foi interrompido logo que todos os clientes com maior prioridade tenham sido servidos. O atraso médio dos clientes no sistema correspondente à classe k é dado

$$\text{por: } W_k = \frac{\left(1/\mu_k\right)(1-\rho_1-\ldots-\rho_k)+(1/2)\sum_{i=1}^{k}\lambda_i E(S_i^{2})}{(1-\rho_1-\ldots-\rho_{k-1})(1-\rho_1-\ldots-\rho_k)},$$

2.38

Note-se que o atraso médio da classe *k*, ao contrário do sistema com disciplina de serviço não-preemptiva, não depende das classes *k+1* a *n*.

2.3.4.1 Exemplo de sistema M/G/1 com prioridades

Como exemplo de utilização de modelo M/G/1 com prioridade, pode-se também, utilizar o mesmo exemplo utilizado para o caso do modelo M/M/1 considerando que a ligação é partilhada por fluxos de pacotes de dois utilizadores. Cada utilizador gera pacotes com comprimento fixo igual a 500bits. Pretende-se que um dos utilizadores tenha um atraso médio inferior a 100*ms* e o outro utilizador um atraso médio inferior a 260*ms*.

Esta especificação não pode ser cumprida se os fluxos de ambos os utilizadores forem misturados num mesmo buffer e servidos de acordo com uma disciplina FIFO (First-In-First-Out).

Trata-se de um sistema M/D/1. A taxa na qual os pacotes são oferecidos (λ) é 16 pacotes/*s*. A taxa de serviço (μ) é 20 pacotes/*s*. Portanto, a taxa de utilização é igual a 0.8. O atraso médio será:

$$W = \frac{\rho/2\mu}{(1-\rho)} + \frac{1}{\mu} = 150\,ms$$

O que não cumpre a especificação do utilizador que necessita de um menor atraso médio.

Uma vez que deve haver partilha de recursos entre os dois fluxos, não é aceitável uma solução baseada em multiplexagem temporal ou multiplexagem de frequência. Pode usar-se prioritização estrita. Neste caso o fluxo com maior prioridade teria um atraso médio dado por:

$$W_1 = \frac{\rho / 2\mu}{(1-\rho_1)} + \frac{1}{\mu} = 83{,}3\, ms$$

E o de menor prioridade

$$W_2 = \frac{\rho / 2\mu}{(1-\rho_1)(1-\rho_1-\rho_2)} + \frac{1}{\mu} = 216{,}6\, ms$$

Tabela 2.3 - Resultados do exemplo do sistema M/G/1 com prioridade

Atraso médio no sistema	**Valores Calculados**	**Valores Simulados (8 simulações)**										
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Média**	**Variância**	**Intervalo de confiança para 90%**
W – (Fluxos misturados)	0,15	0,1567	0,1433	0,1442	0,1497	0,1496	0,1496	0,162	0,1457	0,1501	0,00004	[0.1464,0.1538]
W_1 - (fluxo de maior prioridade)	0,0833	0,0886	0,0821	0,0835	0,0814	0,0811	0,09	0,0873	0,0814	0,0844	0,000013	[0.0823,0.0865]
W_2- (fluxo de menor prioridade)	0,2166	0,2457	0,184	0,2376	0,2153	0,1962	0,2582	0,2921	0,1935	0,2278	0,001389	[0.2061,0.2495]

No capítulo seguinte, apresentam-se os algoritmos necessários a obtenção dos valores apresentados nas tabelas 2.1, 2.2 e 2.3.

CAPÍTULO III - SIMULAÇÃO DE LIGAÇÕES PONTO-A-PONTO

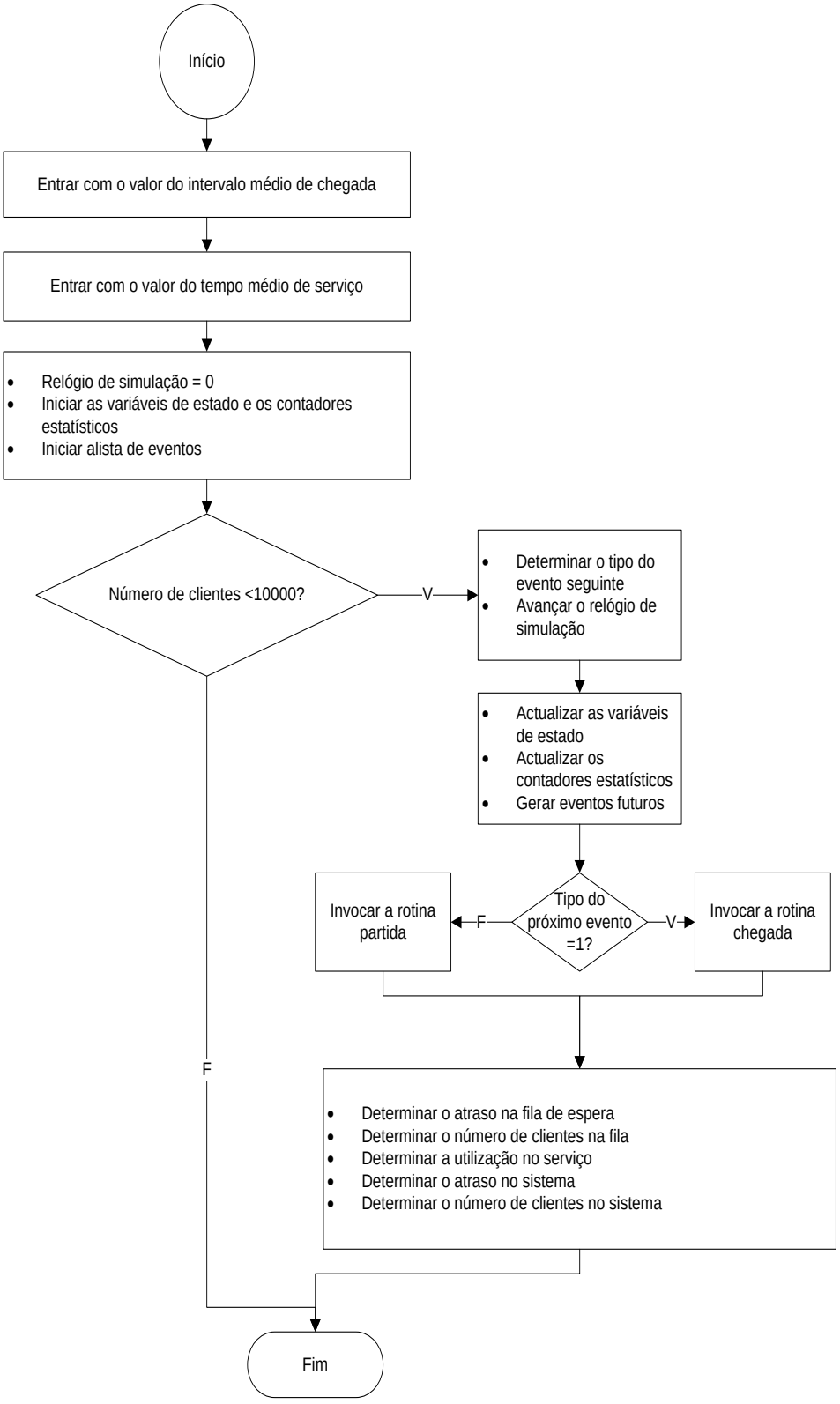
Introdução

Como anteriormente dito, este capítulo destina-se à descrição da implementação dos algoritmos que irão servir de suporte para a simulação e confirmação dos resultados analíticos encontrados no capítulo anterior.

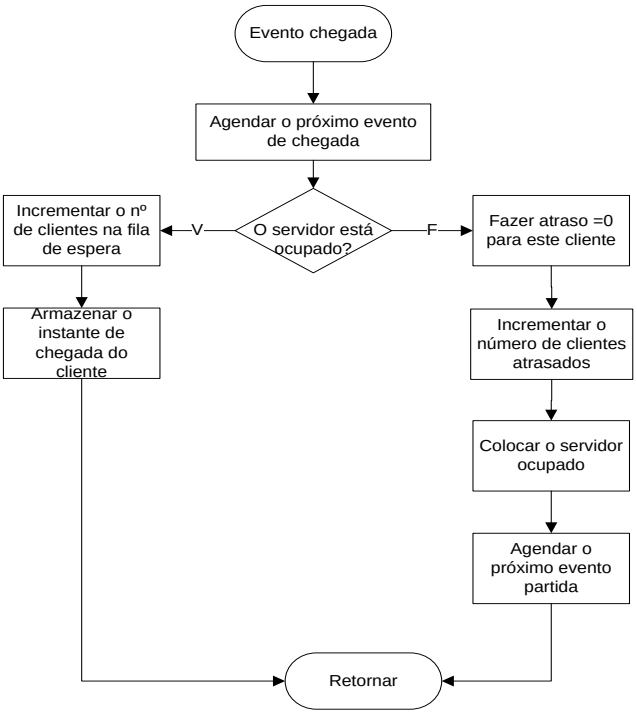
Apresentam-se aqui, os fluxogramas, as rotinas e as suas descrições relativos aos modelos M/M/1 e M/G/1 com prioridade. Por existir uma grande semelhança entre os sistemas M/M/1 e M/D/1, apresenta-se para o caso do sistema M/M/1, apenas a rotina. O mesmo acontece com os sistemas M/G/1 com e sem prioridade. Para passar de um sistema com prioridade para um sem prioridade, basta atribuir o mesmo grau de prioridade para os fluxos.

3.1 Sistemas M/M/1 e M/D/1

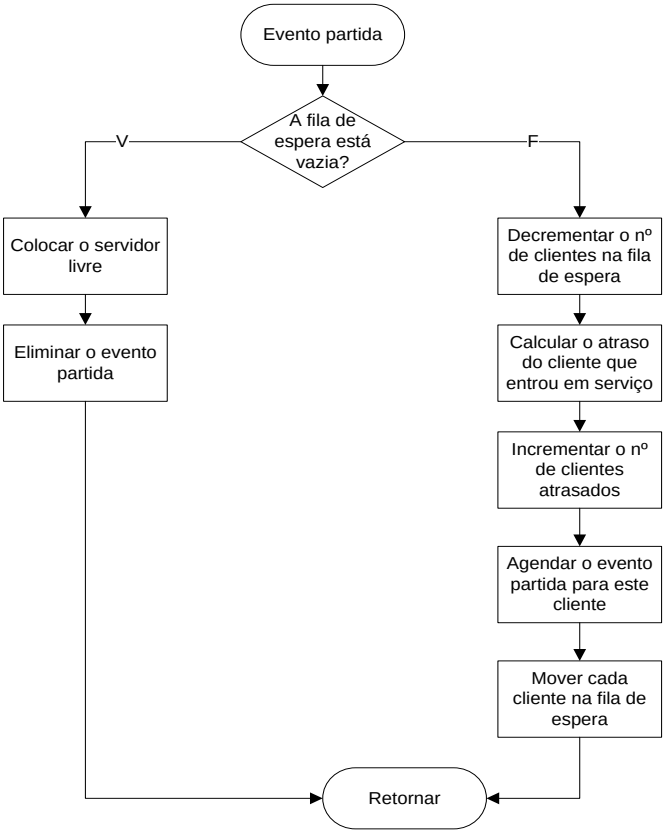
3.1.1 Fluxograma do programa principal



3.1.2 Fluxograma da rotina do evento chegada



3.1.3 Fluxograma da rotina do evento partida



3.1.4 Código em MATLAB do sistema M/M/1

```
MeanInterarrival = input('Entrar com o valor de meaninterarrival =');
MeanService = input('Entrar com o valor de meanservice = ');
Time=0;
ServerStatus=0;
NumInQ=0;
TimeLastEvent=0;
NumCustsDelayed=0;
TotalOfDelays=0;
AreaNumInQ=0;
AreaServerStatus=0;
TimeNextEvent (1) =Time+exprnd (MeanInterarrival);
TimeNextEvent (2) =Inf;
while NumCustsDelayed<10000
%Timing2;

    [Time, NextEventType] =min (TimeNextEvent);

%Update;
    TimeSinceLastEvent=Time-TimeLastEvent;
    TimeLastEvent=Time;
    AreaNumInQ=AreaNumInQ+NumInQ*TimeSinceLastEvent;

      AreaServerStatus=AreaServerStatus+ServerStatus*TimeSinceLastEven
      t;
    if NextEventType==1

%Arrive;
           TimeNextEvent(1)=Time+exprnd(MeanInterarrival);
        if ServerStatus==1% Se servidor ocupado
            NumInQ=NumInQ+1;
            TimeArrival(NumInQ)=Time;
        else
            NumCustsDelayed=NumCustsDelayed+1;
            ServerStatus=1;%OCUPADO
            TimeNextEvent(2)=Time+exprnd(MeanService);
        end
    else

%Depart
```

```
        if NumInQ==0% FILA VAZIA
            ServerStatus=0;
            TimeNextEvent(2)=Inf;
        else
            NumInQ=NumInQ-1;
            Delay=Time-TimeArrival(1);
            TotalOfDelays=TotalOfDelays+Delay;
            NumCustsDelayed=NumCustsDelayed+1;
            TimeNextEvent(2)=Time+exprnd(MeanService);
            TimeArrival=TimeArrival(2:length(TimeArrival));
        end
    end
end

%Report;

    AvgDelayInQ=TotalOfDelays/NumCustsDelayed
    AvgNumInQ=AreaNumInQ/Time
    ServerUtilization=AreaServerStatus/Time
    AvgDelayInSistm=AvgDelayInQ+MeanService
    AvgNumInSistm=AreaNumInQ/Time+AreaServerStatus/Time
```

3.1.4.1 Descrição

No programa começa-se por dar entrada aos parâmetros e iniciar as variáveis de estado. Enquanto existir clientes para entrar no sistema, o simulador determina o tipo do próximo evento, avança o relógio da simulação e actualiza as variáveis de estado bem como os contadores estatísticos. Se o tipo do próximo evento for 1 (um) o simulador agenda o próximo evento de chegada. Nesta condição, se o servidor estiver ocupado o simulador incrementa o número de cliente na fila de espera e armazena o instante de chegada deste cliente. Caso o servidor estiver livre, o cliente será instantaneamente atendido e portanto o atraso para este cliente será igual a zero. O simulador passa a incrementar o número de clientes atrasados, coloca o servidor no estado ocupado e agenda o próximo evento de partida. Na partida, se a fila estiver vazia o servidor é colocado no estado livre e elimina o evento de partida. Caso contrário, o simulador

retira clientes da fila e calcula o atraso para o cliente que entrou em serviço. Por outro lado, incrementa o número de clientes atrasados e agenda a partida para o cliente em serviço ao mesmo tempo que move cada cliente na fila de espera. Finalmente o simulador determina: o atraso na fila de espera; o número de clientes na fila; a utilização no serviço; o atraso no sistema e o número de clientes no sistema.

3.1.5 Código em MATLAB para o sistema M/D/1

```
MeanInterarrival = input('Entrar com o valor de meaninterarrival =');
MeanService = input('Entrar com o valor de meanservice = ');

%init;
Time=0;
ServerStatus=0;
NumInQ=0;
TimeLastEvent=0;

NumCustsDelayed=0;
TotalOfDelays=0;
AreaNumInQ=0;
AreaServerStatus=0;

TimeNextEvent(1)=Time+exprnd(MeanInterarrival);
TimeNextEvent(2)=Inf;
while NumCustsDelayed<10000

%Timing2;

    [Time,NextEventType]=min(TimeNextEvent);
    %Update;
    TimeSinceLastEvent=Time-TimeLastEvent;
    TimeLastEvent=Time;
    AreaNumInQ=AreaNumInQ+NumInQ*TimeSinceLastEvent;
    AreaServerStatus=AreaServerStatus+
    ServerStatus*TimeSinceLastEvent;

    if NextEventType==1
```

```
%Arrive;

        TimeNextEvent(1)=Time+exprnd(MeanInterarrival);
        if ServerStatus==1% SE SERVIDOR ESTÁ VAZIO
            NumInQ=NumInQ+1;
            TimeArrival(NumInQ)=Time;
        else
            NumCustsDelayed=NumCustsDelayed+1;
            ServerStatus=1;%OCUPADO
            TimeNextEvent(2)=Time+MeanService;
        end
    else

 %Depart

        if NumInQ==0% FILA VAZIA
            ServerStatus=0;
            TimeNextEvent(2)=Inf;
        else
            NumInQ=NumInQ-1;
            Delay=Time-TimeArrival(1);
            TotalOfDelays=TotalOfDelays+Delay;
            NumCustsDelayed=NumCustsDelayed+1;
            TimeNextEvent(2)=Time+MeanService;
           TimeArrival=TimeArrival(2:length(TimeArrival));

        end
    end
end

%Report;

    AvgDelayInQ=TotalOfDelays/NumCustsDelayed % Atraso na fila
    AvgNumInQ=AreaNumInQ/Time %Número de clientes na fila
    ServerUtilization=AreaServerStatus/Time % utilização do serviço
    AvgDelayInSistm=AvgDelayInQ+MeanService% Atraso no sistema
    AvgNumInSistm=AreaNumInQ/Time+AreaServerStatus/Time % Número de
clientes no sistema
```

3.1.5.1 Descrição

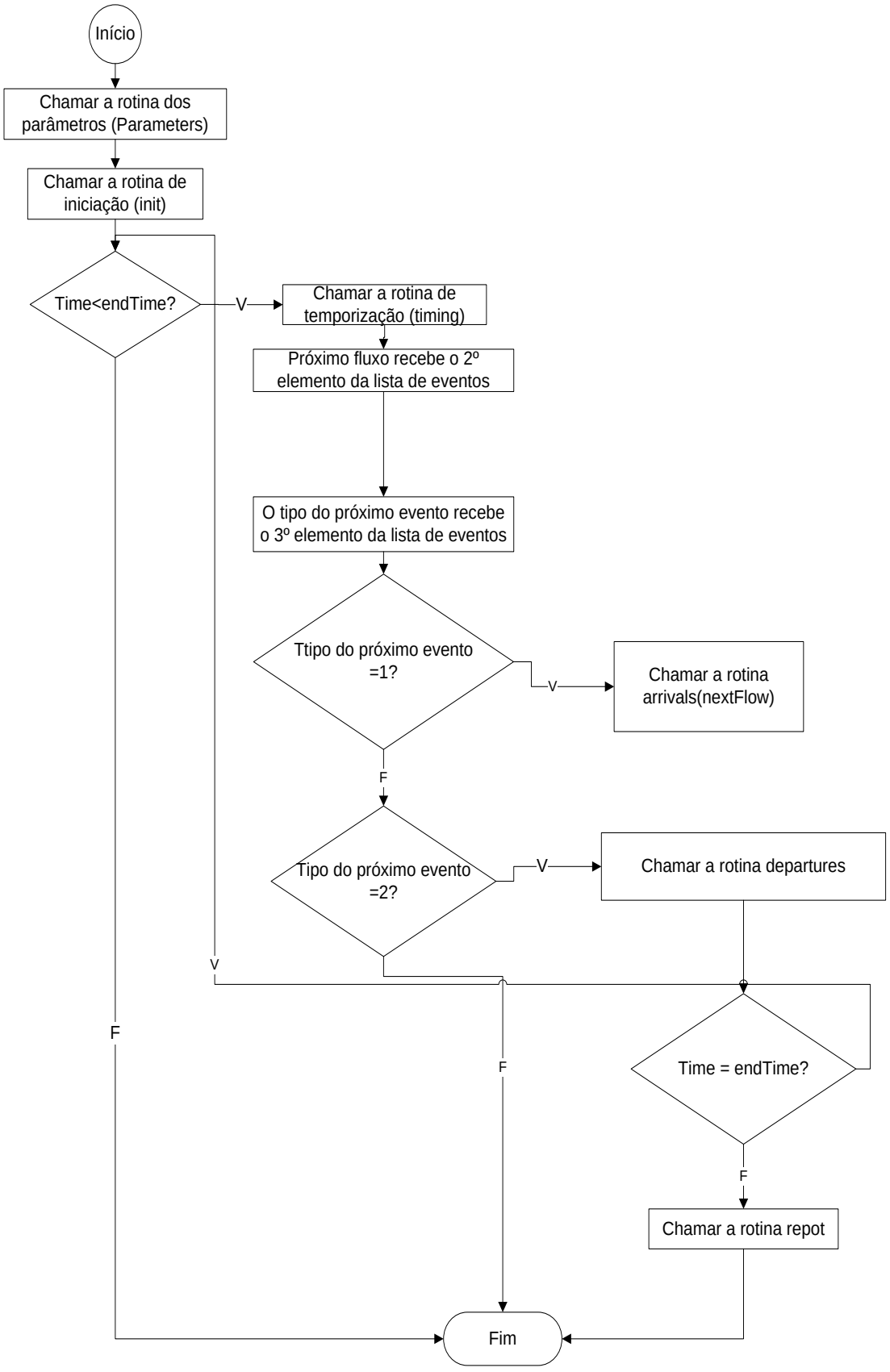
A estrutura de dados para o simulador M/D/1 é idêntica á do sistema M/M/1, uma vez que a diferença entre os dois sistemas está no facto de que no sistema M/D/1 o tamanho dos pacotes é constante, o tempo de serviço dos pacotes é determinístico.

3.2 Algoritmos para o sistema M/G/1 com e sem prioridade

3.2.1 Código em MATLAB do programa principal (mg1cp)

```
function mg1cp

clear all;

global Time;
global endTime;

parameters; %Definition of input parameters

init; %Initialization of data structures

%Main program
while Time < endTime
    [nextEvent]=timing; %Next event
    nextFlow=nextEvent(2); %Flow of next event
    nextEventType=nextEvent(3); %Type of next event
    if nextEventType==1 %If next event is an arrival
        arrivals(nextFlow); %Call arrivals routine
    else %If next event is a departure
        departures; %Call departures routine
    end
end

report; %Computes the performance metrics and writes the simulation
results
```

3.2.1.1 Fluxograma do programa principal



3.2.1.2 Descrição

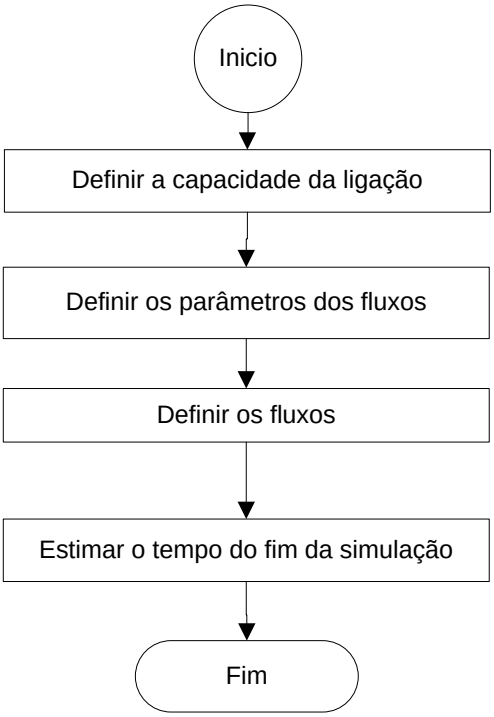
Ao ser chamada, procura as informações contidas na rotina parameters onde estão definidas os parâmetros de entrada, tais como a capacidade da ligação, as características dos fluxos, etc. Em seguida chama a rotina init para a inicialização da estrutura de dados.

O programa principal inserido nesta rotina estabelece a comparação entre o tempo real da simulação e o tempo estimado para o fim da simulação (normalmente é definido na rotina parameters). Enquanto o tempo real da simulação for inferior ao tempo estimado para a simulação, a rotina timing é chamada. A rotina actualiza o tempo, procura o fluxo do próximo evento e o tipo do próximo evento. Se o tipo do próximo evento é a chegada de pacote, a rotina arrivals(nextFlow) é chamada, senão a rotina departures será chamada. Depois deste processo a rotina report é chamada para efectuar os cálculos necessários e solicitados.

3.2.2 Rotina parameters

```
function parameters

global LinkCapacity;
global Flows;
global endTime;

LinkCapacity=10000;
Flows={[2,1/8,500,1];[2,1/8,500,2]};
      [Flows is a cell array where each cell corresponds to a
      flow, and each flow is a vector with 4 elements
      corresponding to (1) source type, (2) mean interarrival
      time (in seconds), (3) the mean packet length (in bits),
      and(4) priority level. There are two types of sources: 1 =
      Poisson arrivals and exponentially distributed sizes; 2 =
      Poisson arrivals and fixed sizes.The levels of priority
      must be consecutive integers starting at 1, where a lower
      number corresponds to a higher priority].

EndTime=1000* (1/8);
```

3.2.2.1 Fluxograma da rotina parameters



3.2.2.2 Descrição

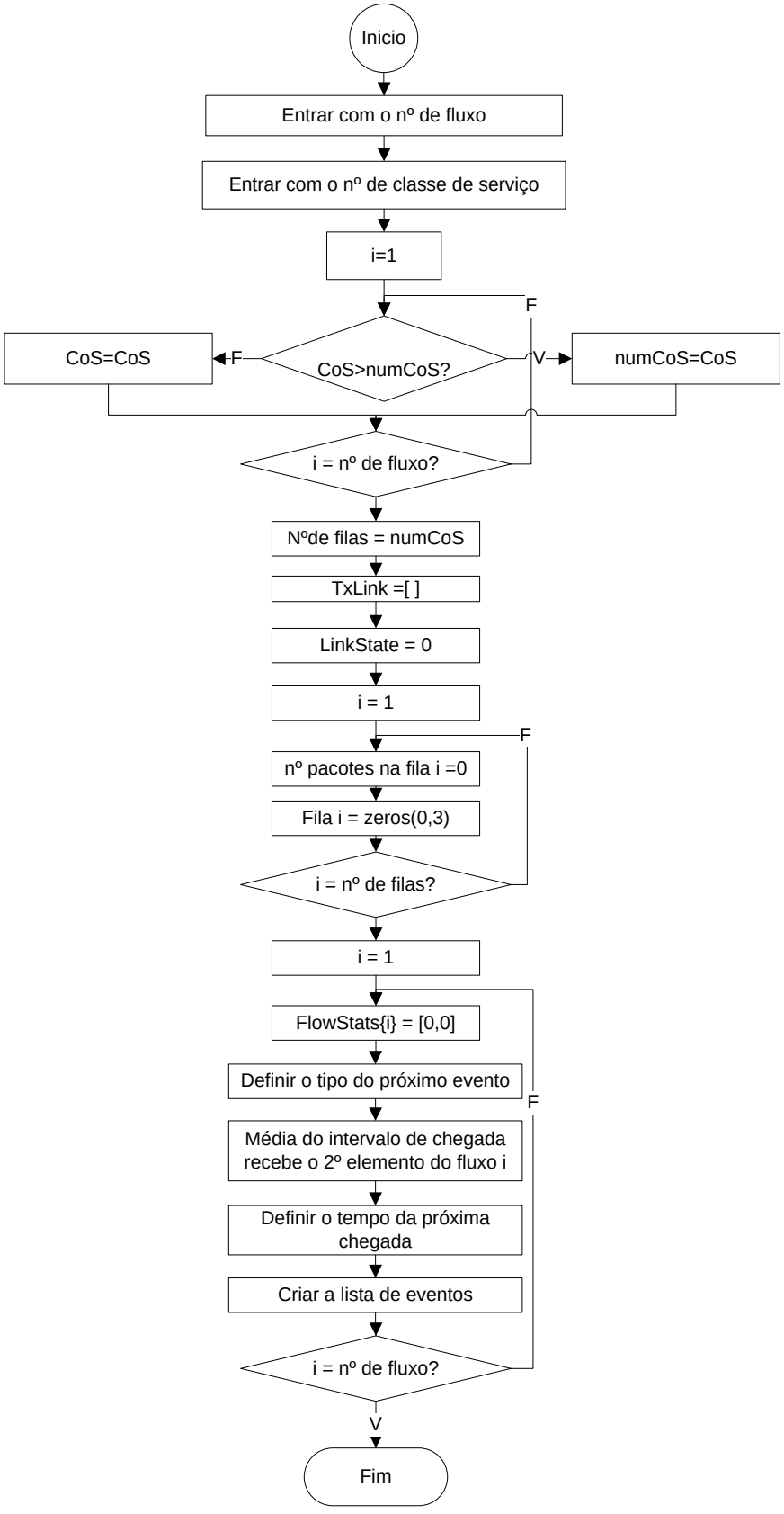
Para efectuar as simulações torna-se necessário definir os paramentros de entrada. Para o efeito, esta rotina define a capacidade da ligação, define os parâmetros dos fluxos e o tempo aproximado para o fim das simulações.

3.2.3 Rotina init

```
function init

global Time;
global EventList;
global Flows;
global numFlows;
global FlowStats;
global Queues;
global numQueues;
global numPacketsInQueues;
global TxLink;
global LinkState;

%Initialization of simulation clock
Time=0;

%Number of priority levels at the link
numFlows=size(Flows,1);
```

```
numCoS=1;
for i=1:numFlows
    if Flows{i}(4)>numCoS
        numCoS=Flows{i}(4);
    end
end

%Initialization of link data structures
numQueues=numCoS; %One queue for each priority level
TxLink=[]; %Transmission link is empty
LinkState=0; %State of transmission link is idle
for i=1:numQueues
    numPacketsInQueues(i)=0; %This queue is empty
    Queues{i}=zeros(0,3); %Initialization of Queues
end

%Initialization of flow data structures
for i=1:numFlows
    FlowStats{i}=[0,0,0]; %Statistics of this flow
    nextEventType=1; %Next event type is arrival
    MeanInterarrival=Flows{i}(2); %Mean interarrival time of
this flow
    nextArrivalTime=exprnd(MeanInterarrival); %Arrival time of
next packet of this flow
    EventList(i,:)=[nextArrivalTime i nextEventType]; %Schedules
next packet arrival for this flow
end
```

3.2.3.1 Fluxograma da rotina init



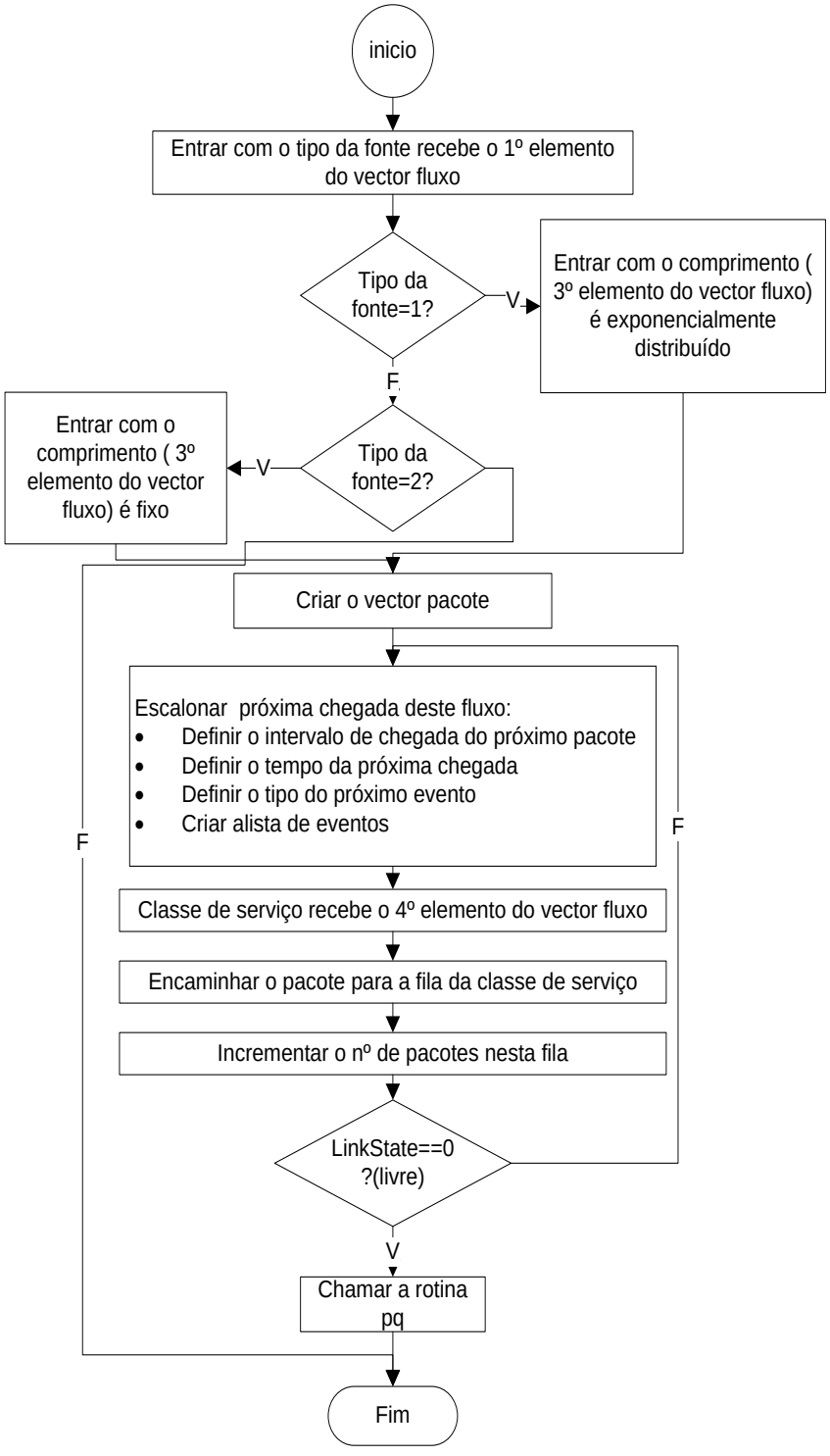
3.2.3.2 Descrição

Esta é uma rotina de inicialização das estruturas de dados. Ao ser chamada, inicializa o relógio de simulação, define o número do fluxo, bem como o número das classes de serviço. A classe de serviço é definida em função do número de classe de serviço. A rotina define uma fila para cada nível de prioridade, coloca a ligação de transmissão em vazio, coloca a ligação no estado livre, inicializa as filas e coloca-as no estado vazio. Inicializa os contadores estatísticos para os fluxos e os eventos.

3.2.4 Rotina arrivals

```
function arrivals(thisFlow)

global Time;
global EventList;
global Flows;
global Queues;
global numPacketsInQueues;
global LinkState;

%Generates this packet
SourceType=Flows{thisFlow}(1); %Source type of this flow
if SourceType==1 %If source type is 1
    thisLength=exprnd(Flows{thisFlow}(3)); %Packet size is
exponentially distributed
else %If source type is 2
    thisLength=Flows{thisFlow}(3); %Packet size is fixed
end
thisPacket=[thisFlow Time thisLength]; %Builds this packet

%Schedules next arrival of this flow
nextInterarrival=exprnd(Flows{thisFlow}(2)); %Interarrival for
next packet
nextArrivalTime=Time+nextInterarrival; %Time of next arrival
nextEventType=1; %Next event type is arrival
EventList(end+1,:)=[nextArrivalTime thisFlow nextEventType];
%Places arrival event in event list

%Enques this packet
thisCoS=Flows{thisFlow}(4);
Queues{thisCoS}(end+1,:)=thisPacket;
numPacketsInQueues(thisCoS)=numPacketsInQueues(thisCoS)+1;

%Calls packet scheduler if link is idle
if LinkState==0
    pq;
end
```

3.2.4.1 Fluxograma da rotina arrivals



3.2.4.2 Descrição

Esta rotina gera chegada de pacotes para os fluxos. Gera pacotes, encaminha-os para as filas e chama o escalonador de pacotes. Aos pacotes são atribuídos: o número dos

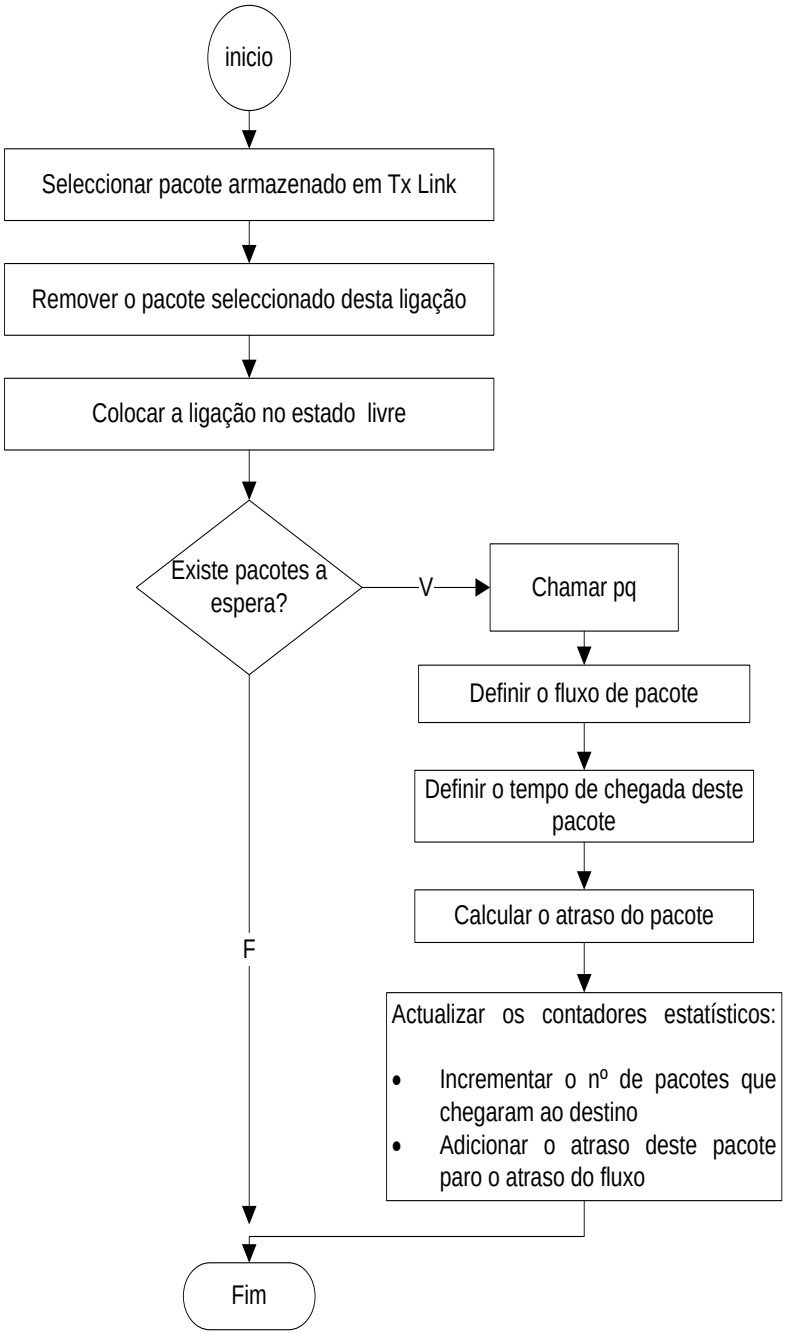
fluxos, o tempo de chegada e o comprimento que são armazenados em um vector - thisPacket=[thisFlow Time thisLength].

Define o tipo da fonte do fluxo. Se a fonte for do tipo 1, o comprimento do pacote é exponencialmente distribuído; se for do tipo 2 o comprimento do pacote será fixo. Escalona o próximo evento de chegada e cria uma lista de eventos. Encaminha o pacote para a fila prioritária e incrementa o número de pacotes nesta fila. Chama o escalonador de pacotes pq, se a ligação estiver livre.

3.2.5 Rotina departures

```
function departures

global TxLink;
global LinkState;
global Time;
global FlowStats;
global numPacketsInQueues;
global numQueues;

thisPacket=TxLink; %Packet that ended transmission
%Removes this packet from the link
TxLink=[];
LinkState=0; %Set link state to idle

%Calls packet scheduler if there are packets waiting
if not(isequal(numPacketsInQueues,zeros(1,numQueues)))
    pq;
end

%Calculates the delay of this packet
thisFlow=thisPacket(1); %Flow of this packet
thisArrivalTime=thisPacket(2); %Arrival time of this packet
thisPacketDelay=Time-thisArrivalTime; %Calculates packet delay

%Updates statistical counters
thisLength=thisPacket(3);
FlowStats{thisFlow}(1)=FlowStats{thisFlow}(1)+1; %Increments
number of packets that arrived at destination
FlowStats{thisFlow}(2)=FlowStats{thisFlow}(2)+thisPacketDelay;
%Adds delay of this packet to flow delay
```

3.2.5.1 Fluxograma da rotina departures



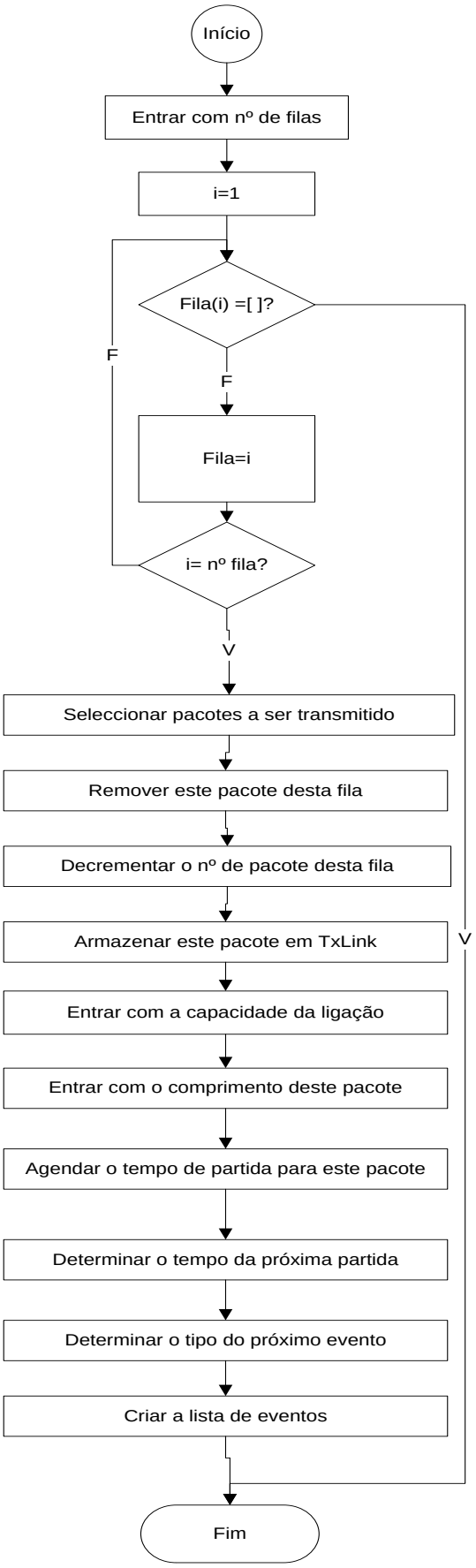
3.2.5.2 Descrição

Esta rotina processa partida de pacotes para uma ligação, remove o pacote de TxLink, coloca a ligação no estado livre e chama o escalonador de pacotes se haver pacotes a espera. Determina o atraso dos pacotes que entraram em serviço e actualiza os contadores estatísticos.

3.2.6 Rotina pq

```
function pq

global Queues;
global numPacketsInQueues;
global TxLink;
global LinkState;
global LinkCapacity;
global EventList;
global Time;
global numQueues;

%Selects highest priority queue with enqueued packets
thisQueue=0;
for i=1:numQueues
    if isempty(Queues{i})
        %do nothing
    else
        thisQueue=i;
        break;
    end
end

%Transfers selected packet to transmission link
thisPacket=Queues{thisQueue}(1,:);%Reads packet to be
transmitted
Queues{thisQueue}(1,:)=[]; %Removes selected packet from this
queue
numPacketsInQueues(thisQueue)=numPacketsInQueues(thisQueue)-1;
%Decrements number of packets in this queue
TxLink=thisPacket; %Stores this packet at the transmission link
LinkState=1; %Set state of link to busy

%Schedules departure of this packet
thisLength=thisPacket(3); %Length of this packet
nextDepartureTime=Time+thisLength/LinkCapacity; %Time of next
departure
nextEventType=2; %Next event type is departure
EventList(end+1,:)=[nextDepartureTime 0 nextEventType]; %Places
departure event in event list
```

3.2.6.1 Fluxograma da rotina pq



3.2.6.2 Descrição

Por se tratar de um escalonador de prioridades, utiliza-se várias filas sendo que cada uma com um nível de prioridade definida. A rotina selecciona o próximo pacote a ser transmitido, (thisPacket=Queues{thisLink,thisQueue}(1,:)),usando o escalonador prioridade estrita, remove este pacote da fila prioritária para a ligação de transmissão - Queues{thisLink,thisQueue}(1,:)=[ ]. O pacote removido será armazenado na rotina TxLinks{thisLink} e a sua partida é programada. Para isso, são determinados: o comprimento do pacote, a capacidade da ligação, o tempo da próxima partida e o tipo do próximo evento (neste caso é uma partida). A rotina cria ainda um espaço chamado Lista de Eventos, onde os parâmetros de partida são guardados e organizados.

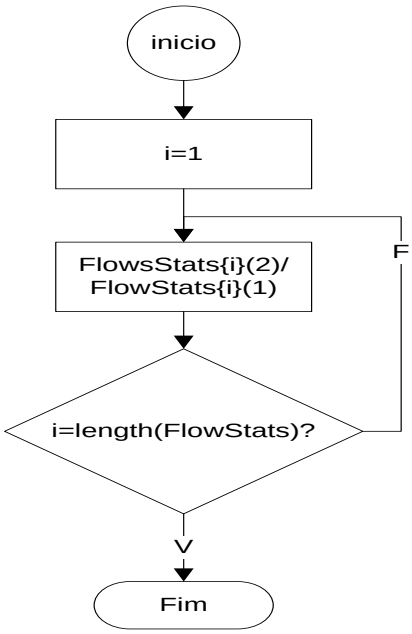
Após a partida de um pacote uma fila vazia pode se formar. Se todas as filas estiverem vazias o escalonador não faz nada.

Esta rotina permite fazer a diferenciação da qualidade de serviço, assunto tratado no último capítulo desta dissertação.

3.2.7 Rotina report

```
function report

global FlowStats;

for i=1:length(FlowStats)
    disp('Average delay in flow');
    disp(i);
    disp(' = ');
    disp(FlowStats{i}(2)/FlowStats{i}(1));
end
```

3.2.7.1 Fluxograma da rotina report



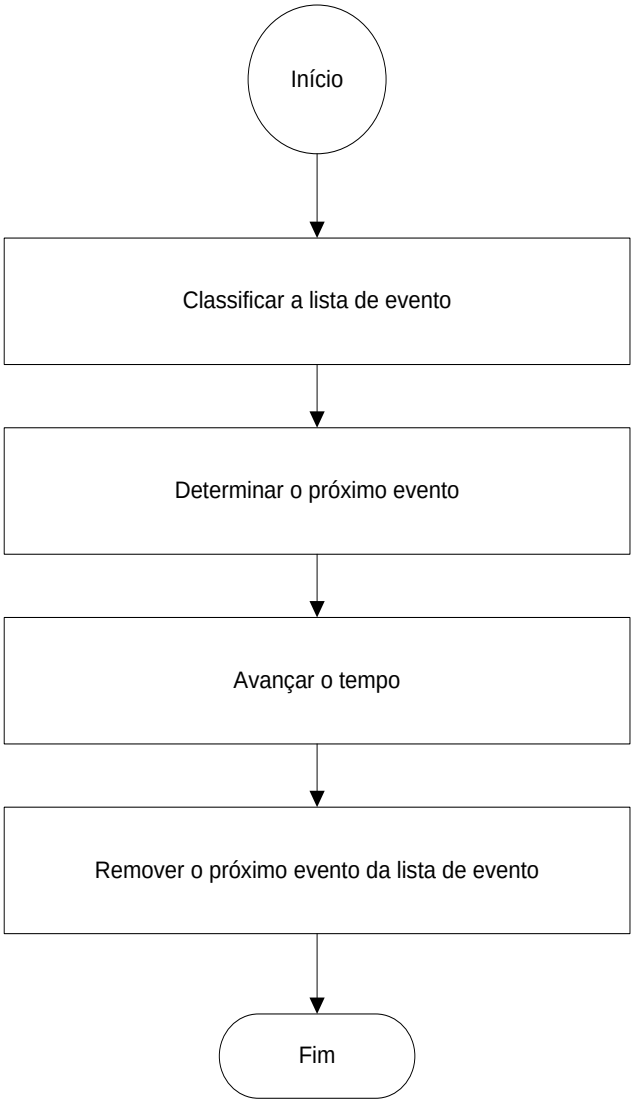
3.2.7.2 Descrição

Esta rotina faz a cálculo das solicitações e escreve os resultados.

3.2.8 Rotina timing

```
function [nextEvent]=timing

global Time;
global EventList;

EventList=sortrows(EventList,1); %Sorts event list
nextEvent=EventList(1,:); %Determines next event
Time=nextEvent(1); %Advances clock
EventList(1,:)=[]; %Removes next event from event list
```

3.2.8.1 Fluxograma da rotina timing



3.2.8.2 Descrição

Os eventos são armazenados numa matriz chamada Lista de Eventos, onde cada fila representa um evento. Os eventos são definidos pelo tempo de ocorrência, pelo número de fluxo e pelo tipo do evento (1- chegada ou 2 – partida). Esta rotina classifica a lista de eventos, determina o próximo evento, avança o relógio da simulação para o tempo do próximo evento e volta para remover este evento da lista de eventos.

CAPÍTULO IV – ANÁLISE DE REDES COM COMUTAÇÃO DE PACOTES E ENCAMINHAMENTO FIXO

Introdução

A comutação de pacotes tem por finalidade a optimização da ocupação das ligações da rede, partindo a mensagem original em fragmentos denominados pacotes, de comprimento fixo ou variável. A cada pacote associa-se informações de controlo. Os pacotes são individualmente lançados para a rede, ficando os equipamentos constituintes da rede com a responsabilidade de os encaminhar de forma mais rápida e eficiente nas ligações até ao destino.

Existem basicamente duas formas distintas para a comutação de mensagens: comutação de pacotes e comutação de circuitos. Nesta última forma, toda a comunicação é feita através de um circuito específico que é integralmente ocupado pelo processo de transmissão de dados. Por outro lado, a comunicação entre dois terminais pode ser estabelecida por encaminhamento fixo - nestas condições, conhece-se previamente o conjunto global das ligações que se deseja estabelecer – ou por encaminhamento dinâmico – quando os pedidos de ligações chegam segundo um processo estocástico e os caminhos óptimos são libertados ao fim de algum tempo.

4.1 Aproximação de Kleinrock

Kleinrock sugere um método para aproximar o número médio de pacotes no sistema e o atraso médio sofrido pelos mesmos. Consideremos uma rede composta por vários

fluxos, onde cada fluxo segue uma rota que por sua vez, consiste numa sequência de ligações entre a origem ao destino. A taxa total de chegada de pacotes a um determinado ramo (*i,j*) é a soma das taxas de chegada de todos os fluxos que passam por esse ramo(caminho entre dois nodos) (KLEINROCK, 1964). Tomando como exemplo a figura 4.1 abaixo, a taxa de chegada de pacotes no ramo (*i, j*) é portanto:

$$\lambda_{(i,j)} = x_2 + x_3 \qquad 4.1$$

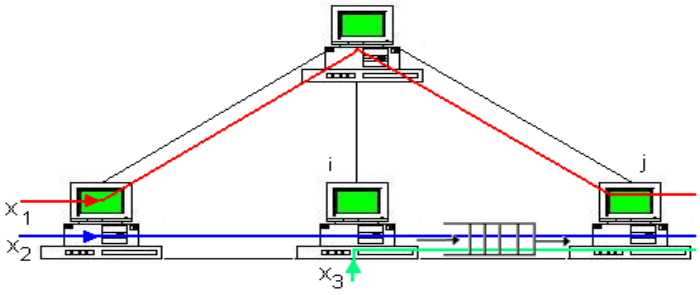


Figura 4.1 - Exemplo de fila em redes de dados

A aproximação de Kleinrock sugere que ao associar vários fluxos de pacotes a um único ramo de transmissão se produz o efeito de restaurar a independência entre os tempos de chegada e o tamanho dos pacotes, e que a utilização do modelo M/M/1 em cada ligação é adequada.

Pode-se aproximar o número médio de pacotes em cada ramo pela equação

$$L_{ij} = \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}} \qquad 4.2$$

que é a equação 2.14 generalizada.

Somando o número médio de pacotes de cada ramo, encontra-se o número médio de pacotes no sistema:

$$L = \sum_{(i,j)} L_{ij} = \sum_{(i,j)} \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}}$$

4.3

Utilizando uma generalização do Teorema de Little, pode-se aproximar o atraso médio de cada pacote no sistema pela seguinte equação:

$$W = \frac{1}{\beta} \sum_{(i,j)} \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}} \qquad 4.4$$

onde β é a soma da taxa de chegada de todos os fluxos que chegam ao sistema.

Com os mesmos argumentos usados para chegar a equação acima, pode-se encontrar o atraso médio que os pacotes de um determinado fluxo levam para atravessar um caminho p:

$$W_p = \sum_{\substack{(i,j) \\ i \neq j}} \left(\frac{\lambda_{ij}}{\mu_{ij}(\mu_{ij} - \lambda_{ij})} + \frac{1}{\mu_{ij}} \right) \qquad 4.5$$

onde (i, j) são os ramos do caminho p.

Para apreciar a dificuldade de obter o valor exacto do atraso de pacotes consideremos o exemplo da figura 4.2. No exemplo da fig. 4.2 com pacotes de comprimento fixo e duas ligações com a mesma capacidade, o atraso na segunda fila de espera é nulo. Neste caso embora a primeira ligação possa ser modelada por um sistema M/D/1, e segunda já não pode.

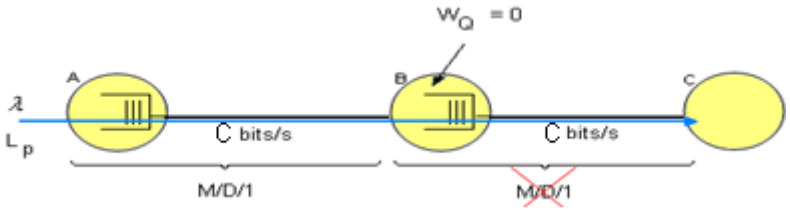


Figura 4.2 – Exemplo da aproximação de Kleinrock

A equação da aproximação de Kleinrock para este caso é:

$$W = \left(\frac{\lambda}{2\frac{C_1}{L_p}\left(\frac{C_1}{L_p} - \lambda\right)} + \frac{1}{\frac{C_1}{L_p}} \right) + \left(\frac{\lambda}{2\frac{C_2}{L_p}\left(\frac{C_2}{L_p} - \lambda\right)} + \frac{1}{\frac{C_2}{L_p}} \right) \qquad 4.7$$

4.2 Exemplo de aplicação da aproximação de Kleinrock.

Como exemplo de aplicação da aproximação de Kleinrock e da equação 4.5 considera-se a rede da figura 4.3:

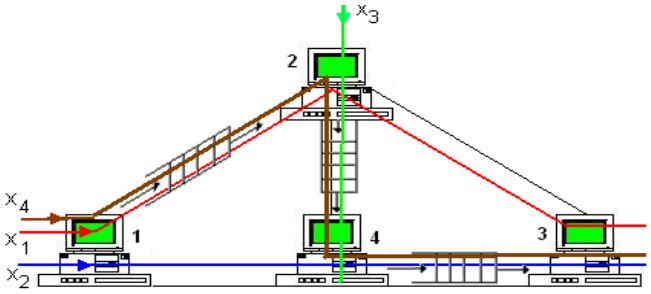


Figura 4.3– Rede para o exemplo de utilização do teorema de Kleinrock

Os tráfegos x_1, x_2 *e* x_3, representados na figura 4.3, são de Poisson com pacotes de tamanho médio igual a 600 bits e cada um tem 10Kbps de taxa média. x_4 é um tráfego com pacotes de tamanho médio igual 400 bits e taxa média igual a 4Kbps.

Cenário 1

Num primeiro momento considera-se que todos os ramos têm a mesma capacidade 32Kbps. Nestas condições, o tamanho médio dos pacotes do tráfego combinado é igual a média ponderada do tamanho dos pacotes dos tráfegos individuais, ou seja:

$$\overline{L_p} = \frac{\sum_i \overline{L_{p\,i}} \lambda_i}{\sum_i \lambda_i} \qquad 4.8$$

A taxa média de chegada de pacotes e a taxa média de serviço de cada ramo são respectivamente.

$$\lambda_{i,j} = \frac{\sum_i \lambda_i}{\overline{L_p}} \qquad 4.9$$

$$\mu_{i,j} = \frac{C}{\overline{L_p}} \qquad 4.10$$

onde $\overline{L_p}$ é o tamanho médio dos pacotes do tráfego combinado, $\overline{L_{pi}}$ e λ_i são respectivamente o tamanho médio dos pacotes e a taxa média dos fluxos individuais e C é a capacidade do ramo.

Relativamente a este exemplo pretende-se calcular o atraso médio dos pacotes do fluxo correspondentes ao circuito [1,2,4,3].

O tráfego x_4 combina - se com o tráfego x_1, x_2 *e* x_3 nos ramos (1,2), (4,3) e (2,4) respectivamente.

Pela equação 4.5, o atraso médio que os pacotes deste fluxo sofrem para atravessar o circuito [1,2,4,3] é:

$$W_p = \left(\frac{\lambda_{12}}{\mu_{12}(\mu_{12}-\lambda_{12})} + \frac{1}{\mu_{12}}\right) + \left(\frac{\lambda_{24}}{\mu_{24}(\mu_{24}-\lambda_{24})} + \frac{1}{\mu_{24}}\right) + \left(\frac{\lambda_{43}}{\mu_{43}(\mu_{43}-\lambda_{43})} + \frac{1}{\mu_{43}}\right)$$

O tamanho médio dos pacotes para o ramo (1,2) é, pela equação 4.8

$$\overline{L_p} = \frac{600 \times 10000 + 400 \times 4000}{1000 + 4000} = 542{,}9 \text{ bits/ pacotes}$$

A taxa média de chegada de pacotes e a taxa média de atendimento são, pelas equações 4.9 e 4.10, $\lambda_{12} = \frac{14000}{542{,}9} = 25{,}8$ pacotes/s e $\mu_{12} = \frac{32000}{542{,}9} = 58{,}9$ pacotes/s respectivamente.

Uma vez que todos os ramos têm a mesma taxa de chegada, a mesma taxa de atendimento, o mesmo tamanho médio de pacotes, a equação

$$W_p = \left(\frac{\lambda_{12}}{\mu_{12}(\mu_{12} - \lambda_{12})} + \frac{1}{\mu_{12}} \right) + \left(\frac{\lambda_{24}}{\mu_{24}(\mu_{24} - \lambda_{24})} + \frac{1}{\mu_{24}} \right) + \left(\frac{\lambda_{43}}{\mu_{43}(\mu_{43} - \lambda_{43})} + \frac{1}{\mu_{43}} \right)$$

se reduz a: $W_p = 3\left(\frac{\lambda_{12}}{\mu_{12}(\mu_{12} - \lambda_{12})} + \frac{1}{\mu_{12}} \right)$

Substituindo os valores de λ_{12} e μ_{12}, tem-se:

$W_p = 0{,}09$

Portanto, o atraso médio sofrido pelos pacotes do fluxo é aproximadamente igual a 0,09 segundos.

Cenário 2

Se, por exemplo, a taxa média do fluxo x_3 for alterada para 16Kbps, então um novo valor para o atraso médio dos pacotes do tráfego total será encontrado a partir da equação 4.5:

$$W_p = 2\left(\frac{\lambda_{12}}{\mu_{12}(\mu_{12} - \lambda_{12})} + \frac{1}{\mu_{12}} \right) + \left(\frac{\lambda_{24}}{\mu_{24}(\mu_{24} - \lambda_{24})} + \frac{1}{\mu_{24}} \right) = 0{,}149$$

Tabela 2.4 - Resultado do exemplo de aplicação de Kleinrock

O atraso médio dos pacotes fluxo correspondente ao circuito [1,2,4,3].	**Valores analíticos**	**Valores simulados**
Cenário 1	0,09*s*	0,1128*s*
Cenário 2	0,149*s*	0,1478*s*

É importante salientar que a mesma ideia utilizada por Kleinrock para modelar o sistema M/M/1, pode também ser considerada para o sistema M /G /1, com apenas alguns ajustes.

É evidente a limitação da aproximação de Kleinrock, principalmente quando se usam ligações em cascata, para um sistema cujo tráfego é elevado. Neste caso a aproximação pode ser má.

Para exemplificar este facto, considere uma rede com duas ligações ponto-a-ponto [figura 4.3] de 64 kb/*s* em cascata. As ligações são atravessadas por um fluxo de pacotes caracterizado por chegadas de Poisson com intervalo médio entre chegadas de 1ms e pacotes com comprimento fixo igual a 2 octetos. O percurso do fluxo é ABC. Pretende-se calcular o atraso médio por pacote (a) sem recorrer a aproximações e (b) por aproximação.

a) Cálculo do atraso médio sem recorrer a aproximação.

O tempo de transmissão dos pacotes é: $\frac{1}{\mu} = \frac{16}{64000} = 250\mu s$ e $\rho = \frac{\lambda}{\mu} = \frac{10^3}{4 \times 10^3} = 0{,}25$

A ligação AB pode ser modelada por um sistema M/D/1. Assim, pela equação 2.31,

$$W = \frac{\lambda}{2\mu(\mu - \lambda)} + \frac{1}{\mu} = 292\mu s$$

Na ligação BC não há atraso na fila de espera. Portanto, a atraso médio por pacote do fluxo é $W = W_{AC} + W_{BC} = 292 + 250 = 542\mu s$

b) Cálculo do atraso médio por aproximação.

Substituindo os valores dos parâmetros na equação 4.7, tem-se:

$$W = \left(\frac{\lambda}{2\frac{C_1}{L_p}\left(\frac{C_1}{L_p} - \lambda\right)} + \frac{1}{\frac{C_1}{L_p}} \right) + \left(\frac{\lambda}{2\frac{C_2}{L_p}\left(\frac{C_2}{L_p} - \lambda\right)} + \frac{1}{\frac{C_2}{L_p}} \right) = 583\mu s$$

Como era esperado, existe uma diferença significativa entre o valor exacto e o valor obtido por aproximação.

Tabela 2.5 – Resultado do exemplo de aplicação da aproximação de Kleinrock numa ligação em cascata

Atraso médio no sistema	**Valores analíticos**	**Valor simulado**
Sem aproximação	542µs	543,36µs
Por aproximação	583µ	----------

CAPÍTULO V - SIMULAÇÃO DE REDES COM COMUTAÇÃO DE PACOTES E ENCAMINHAMENTO FIXO

Introdução

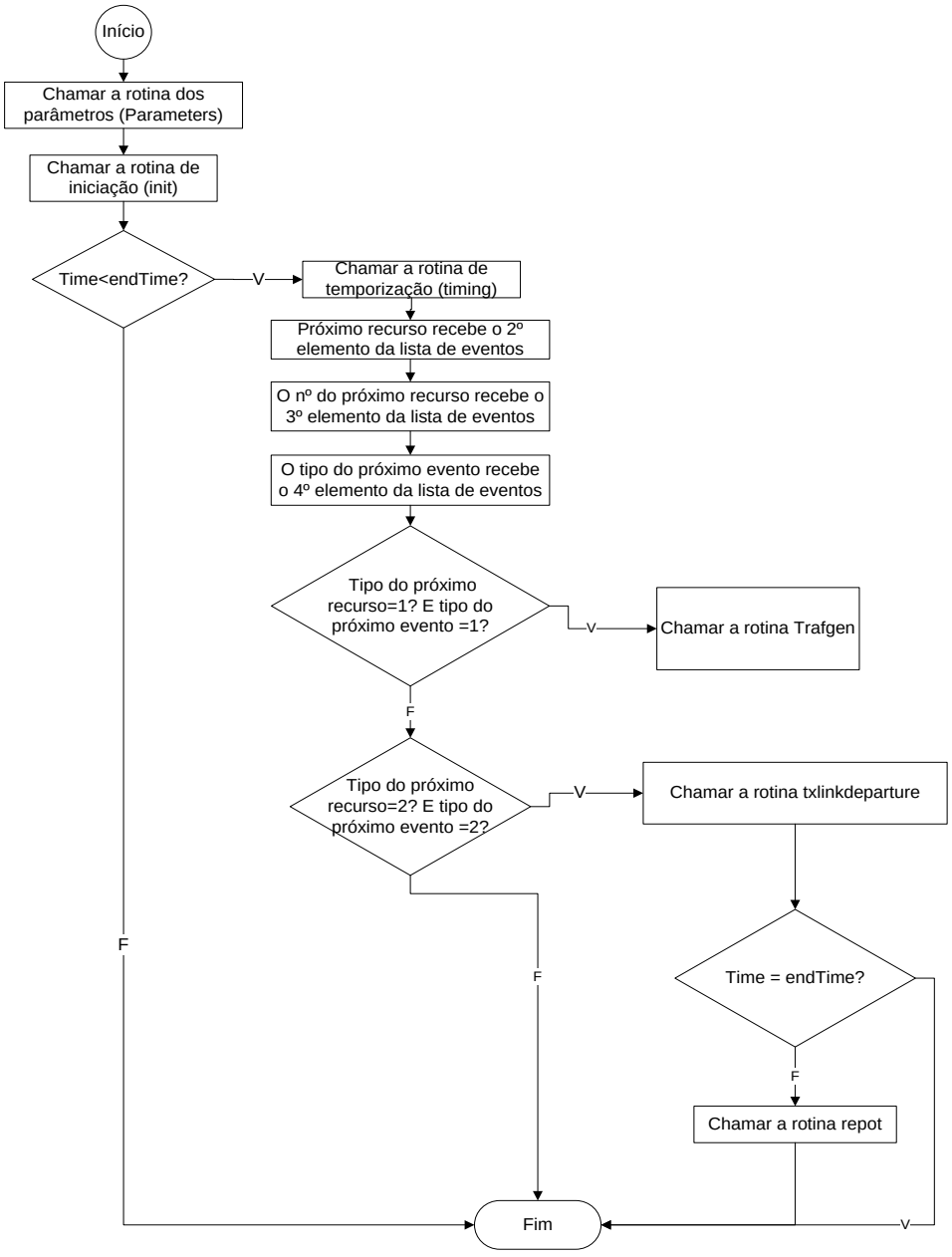
Este capítulo destina-se à descrição da implementação dos algoritmos que irão servir de suporte para a simulação e confirmação dos resultados analíticos encontrados no capítulo quatro, ou seja, apresentam-se aqui, os fluxogramas, as rotinas e descrições relativas aos programas de simulação que irão comprovar a aproximação de Kleinrock, bem como as limitações desta aproximação.

5.1 Rotina pnet (rotina principal)

```
function pnet

clear all;
global Time;
global endTime;

tic;

parameters; %Definition of input parameters

init; %Initialization of data structures

%Main program
while Time < endTime
    [nextEvent]=timing;
    nextResourceType=nextEvent(2);
    nextResourceNumber=nextEvent(3);
    nextEventType=nextEvent(4);
    if (nextResourceType==1&&nextEventType==1)
        trafgen(nextResourceNumber);
    elseif (nextResourceType==2&&nextEventType==2)
        txlinkdeparture(nextResourceNumber);
    end
end

report;
```

```
toc;
```

5.1.1 Fluxograma da rotina pnet



5.1.2 Descrição

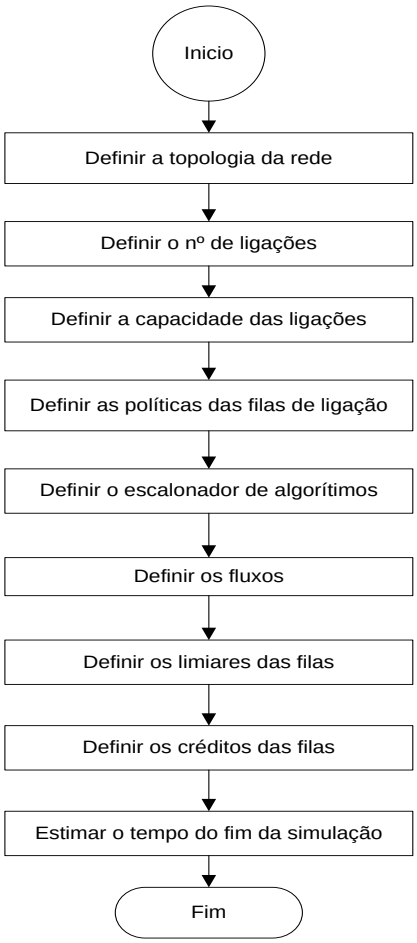
Esta é a rotina principal. Ao ser chamada procura as informações contidas na rotina parameters, onde estão definidas os parâmetros de entrada, tais como: a topologia da rede, as características dos fluxos, etc. Em seguida chama a rotina init para a

inicialização da estrutura de dados. O programa principal inserido nesta rotina estabelece a comparação entre o tempo real da simulação e o tempo estimado para o fim da simulação, sendo que este último normalmente é definido na rotina parameters. Enquanto o tempo real da simulação for inferior ao tempo estimado para a simulação, a rotina actualiza o tempo e procura o tipo e o número do próximo recurso, bem como o tipo do próximo evento. Se o tipo do próximo recurso é a fonte e o tipo do próximo evento é a chegada de pacote, a rotina trafgen é chamada. Se o tipo do próximo recurso for uma ligação e o tipo do próximo evento for uma partida, a rotina txlinkdeparture é chamada. Depois deste processo a rotina report é chamada para efectuar os cálculos necessários e solicitados.

5.2 Rotina parameters

```
function parameters

global LinkCapacity;
global LinkQueuingPolicy;
global LinkSchedulingAlgorithm;
global Topology;
global Flows;
global endTime;
global numLinks;
global DRRThreshold;
global DRRCredit;


%Network topology
Topology=[0 1;
          0 0];
%Links
numLinks=1;
LinkCapacity=[64000]; %Link capacities, in bits/sec
LinkQueuingPolicy=[1]; %Link queuing policy can be 1 -
single queue or 2 - per flow
LinkSchedulingAlgorithm=[1]; %Link scheduling algorithm can
be 1 –fifo or 2 - deficit round-robin
Flows={[1,1/48,2000],[1]}%Flows is a cell array that stores
for each flow the mean interarrival time (in seconds), the
mean packet length (in bits)  and the route.
% End time of simulation
endTime=1000*(1/48);
```

5.2.1 Fluxograma da rotina parameters



5.2.2 Descrição

Nesta rotina são definidas os parâmetros ou as características do sistema, tais como:

- Topologia da rede `(Network topology)` – normalmente é uma matriz que representa a estrutura da rede, permite definir e traçar as rotas de cada fluxo, o número de ligações, etc;
- Número de ligação - O número de ligação é definida de acordo com a matriz de topologia, neste caso o número de ligação é 1, de um ponto a outro, todavia, pode – se construir redes distribuídas, centralizadas ou descentralizadas fazendo algumas modificações na matriz de topologia, conseguindo, portanto sistemas constituído por vários nodos e consequentemente com n ligações;

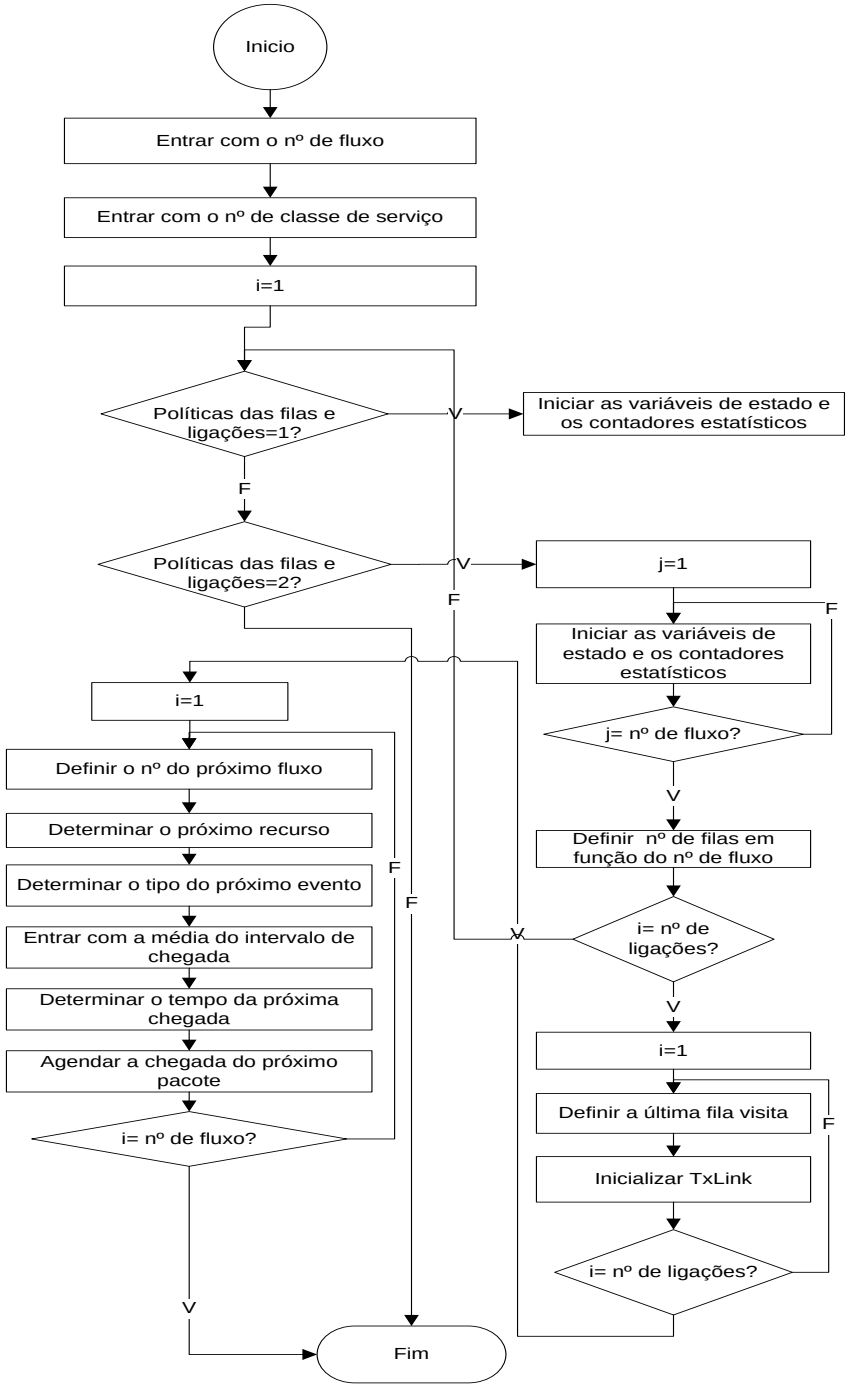
- Capacidade da ligação – é definida por um vector. O número de elementos deste vector depende do número de ligações, ou seja, para cada ligação pode-se definir o valor da capacidade e portanto as ligações podem ter capacidades diferentes;
- Política das filas – de acordo com a opção feita, pode-se seleccionar os diferentes algoritmos de encaminhamento de pacotes, que podem ser: 1 - fifo, 2 – deficit-round-robin;
- Características dos fluxos – Aqui são armazenadas, para cada fluxo, informações como por exemplo: o tipo da fonte; o tempo médio de chegada; o comprimento dos pacotes e a rota;
- Limiares e Créditos - Para o caso da rotina drr (rotina que será usada mais adiante neste trabalho), define-se os limiares de cada fila, bem como os créditos.

5.3 Rotina init

```
function init

global Time;
global EventList;
global Flows;
global FlowStats;
global Queues;
global TxLinks
global LinkQueuingPolicy;
global LinkStats;
global numLinks;
global lastQueue;
global numQueuesVector;

Time=0;
numFlows=size(Flows,1);

%Initialize Queues and LinksStats
for i=1:numLinks
    thisLinkQueuingPolicy=LinkQueuingPolicy(i);
    switch thisLinkQueuingPolicy
        case 1 %single queue
            Queues{i,1}=zeros(0,6);
            LinkStats{i,1}=[0,0];
            numQueuesVector(i)=1;
        case 2 %per flow
            for j=1:numFlows
                Queues{i,j}=zeros(0,6);
                LinkStats{i,j}=[0,0];
            end
            numQueuesVector(i)=numFlows;
    end
end

for i=1:numLinks
    lastQueue(i)=1;
end
```

```
for i=1:numLinks
    TxLinks{i}=[];
end

for i=1:numFlows
    FlowStats{i}=[0,0];
end

for i=1:numFlows
    nextFlow=i; %Number of next flow
    nextResource=1; %Next resource is source
    nextEventType=1; %Next event type is arrival
    MeanInterarrival=Flows{nextFlow,1}(2); %Mean interarrival
time of this flow
    nextArrivalTime=exprnd(MeanInterarrival); %Next arrival time
    EventList(i,:)=[nextArrivalTime nextResource nextFlow
nextEventType]; %Schedules next packet arrival for this flow
end
```

5.3.1 Fluxograma da rotina init



5.3.2 Descrição

Esta rotina destina-se a inicializar as variáveis fundamentais dos programas em questão, tais como: as filas, as ligações, o tempo, os fluxos, etc.

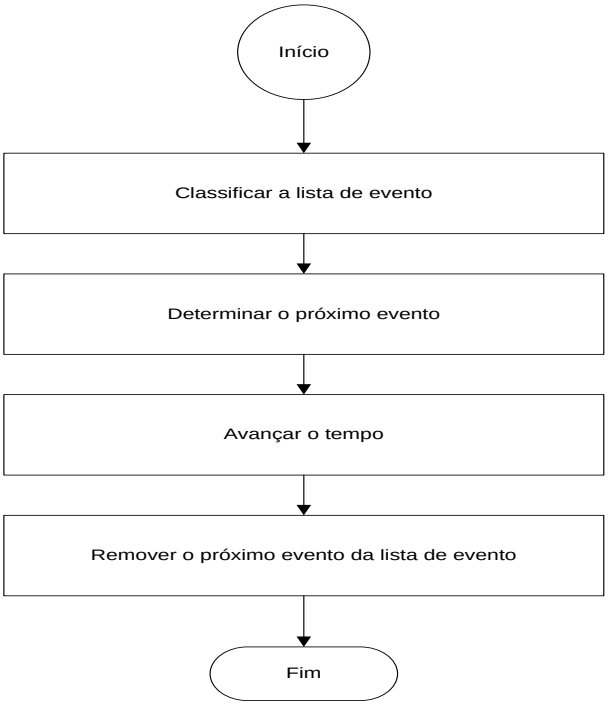
A dimensão das filas, independentemente da política de ligação escolhida - seja a de fila única ou por fluxo - deve ser uma matriz vazia de 0 por 6, uma vez que, as características dos pacotes estão dispostas em vectores numa matriz de uma linha e seis colunas. As variáveis, LinkStats e FlowStats - são vectores 1x2, pelo que as correspondentes matrizes de inicialização têm as mesmas dimensões. TxLinks é normalmente iniciada como sendo uma matriz vazia.

Os parâmetros de chegada também são inicializados para qualquer valor do fluxo, como por exemplo, o próximo recurso, o tipo do próximo evento, o tempo médio de chegada dos fluxos e o tempo da próxima chegada.

5.4 Rotina timing

```
function [nextEvent]=timing

global Time;
global EventList;

EventList=sortrows(EventList,1); %Sorts event list
nextEvent=EventList(1,:); %Determines next event
Time=nextEvent(1); %Advances clock
EventList(1,:)=[]; %Removes next event from event list
```

5.4.1 Fluxograma da rotina timing



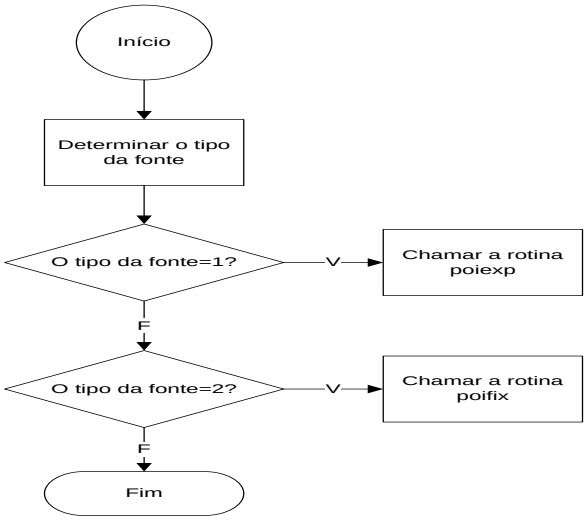
5.4.2 Descrição

Os eventos são armazenados numa matriz chamada lista de eventos (EventList), onde cada fila representa um evento. Os eventos são definidos pelo tempo de ocorrência, tipo de recurso (1 – fonte ou 2 – ligação), número de recursos e pelo tipo de evento (1 – chegada ou 2 – partida). A rotina classifica a lista de eventos EventList=sortrows(EventList,1), determina o próximo evento nextEvent=EventList(1,:), avança o relógio da simulação para o tempo do próximo evento – Time=nextEvent(1), volta para este evento, remove-o da lista de eventos - EventList(1,:)=[ ].

5.5 Rotina trafgen

```
function trafgen(thisFlow)

global Flows;

%Determines the source type
thisSourceType=Flows{thisFlow,1}(1);

switch thisSourceType
    case 1 %Poisson arrivals with exponentially distributed
packet length
        poiexp(thisFlow);
    case 2
        poifix(thisFlow); %Poisson arrivals with fixed
packet length
end
```

5.5.1 Fluxograma da rotina trafgen



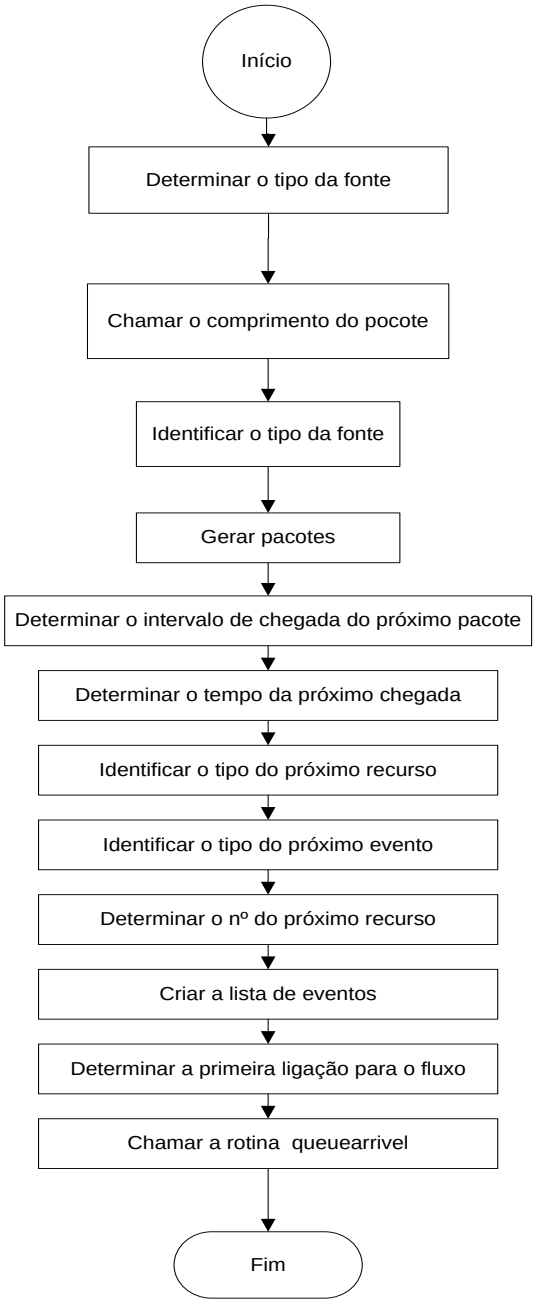
5.5.2 Descrição

Esta rotina determina o tipo da fonte: 1 – processo de chegada de Poisson e pacotes com comprimento exponencialmente distribuídos, ou seja chama a rotina poiexp(thisFlow); 2 – processo de chegada de Poisson e pacotes com comprimento fixo, portanto chama a rotina poifix(thisFlow).

5.6 Rotina poiexp

```
function poiexp(thisFlow)

global Time;
global EventList;
global Flows;

%Generates this packet
thisSourceType=1;

thisLength=exprnd(Flows{thisFlow,1}(3)); %Length of this
packet

thisPacketType=1;
thisPacket=[thisFlow thisSourceType Time thisLength 0
thisPacketType]; %Builds this packet

%Schedules next arrival
nextInterarrival=exprnd(Flows{thisFlow,1}(2)); %Interarrival
for next packet
nextArrivalTime=Time+nextInterarrival; %Time of next arrival
nextResourceType=1; %Next resource is source
nextEventType=1; %Next event type is arrival
nextResourceNumber=thisFlow; %Number of next flow
EventList(end+1,:)=[nextArrivalTime nextResourceType
nextResourceNumber nextEventType]; %Places arrival event in
event list

%Determines first link for this flow
nextLink=Flows{thisFlow,2}(1);

%Delivers this packet to first link
queuearrival(thisPacket,nextLink);
```

5.6.1 Fluxograma da rotina poiexp



5.6.2 Descrição

A rotina mencionada neste tópico gera chegada de pacotes para os fluxos. A fonte gera chegada de Poisson e pacotes com comprimentos exponencialmente distribuídos - thisLength=exprnd(Flows{thisFlow,1}(3)). As informações do pacote gerado - são armazenadas em um vector - thisPacket=[thisFlow thisSourceType Time thisLength 0 thisPacketType]. A eferida rotina programa a próxima chegada de pacotes e cria uma lista de eventos onde as informações referentes a próxima chegada são guardadas - EventList(end+1,:)=[nextArrivalTime nextResourceType nextResourceNumber nextEventType]. O próximo evento normalmente entra na última fila na lista de eventos. Nesta lista constam: intervalo de chegado do próximo pacote, o tempo da próxima chegada, o tipo de recurso (1 - que indica ser uma fonte), o tipo do próximo evento (1 - que indica uma chegada) e o número do próximo fluxo.

A rotina determina ainda, a primeira ligação para o fluxo e encaminha o pacote gerado para esta ligação - queuearrival(thisPacket,nextLink).

5.7 Rotina poifix

```
function poifix(thisFlow)

global Time;
global EventList;
global Flows;

%Generates this packet
thisSourceType=1;

thisLength=Flows{thisFlow,1}(3); %Length of this packet
thisPacketType=1;
thisPacket=[thisFlow thisSourceType Time thisLength 0
thisPacketType]; %Builds this packet

%Schedules next arrival
nextInterarrivalInBucket=exprnd(BucketRate); %Interarrival
for next packet
nextArrivalBucketTime=Time+nextInterarrivalInBucket;
nextInterarrival=exprnd(Flows{thisFlow,1}(2)); %Interarrival
for next packet
nextArrivalTime=Time+nextInterarrival; %Time of next arrival
nextResourceType=1; %Next resource is source
nextEventType=1; %Next event type is arrival
nextResourceNumber=thisFlow; %Number of next flow
EventList(end+1,:)=[nextArrivalTime nextResourceType
nextResourceNumber nextEventType]; %Places arrival event in
event list

%Determines first link for this flow
```

```
nextLink=Flows{thisFlow,2}(1);

%Delivers this packet to first link
queuearrival(thisPacket,nextLink);
```

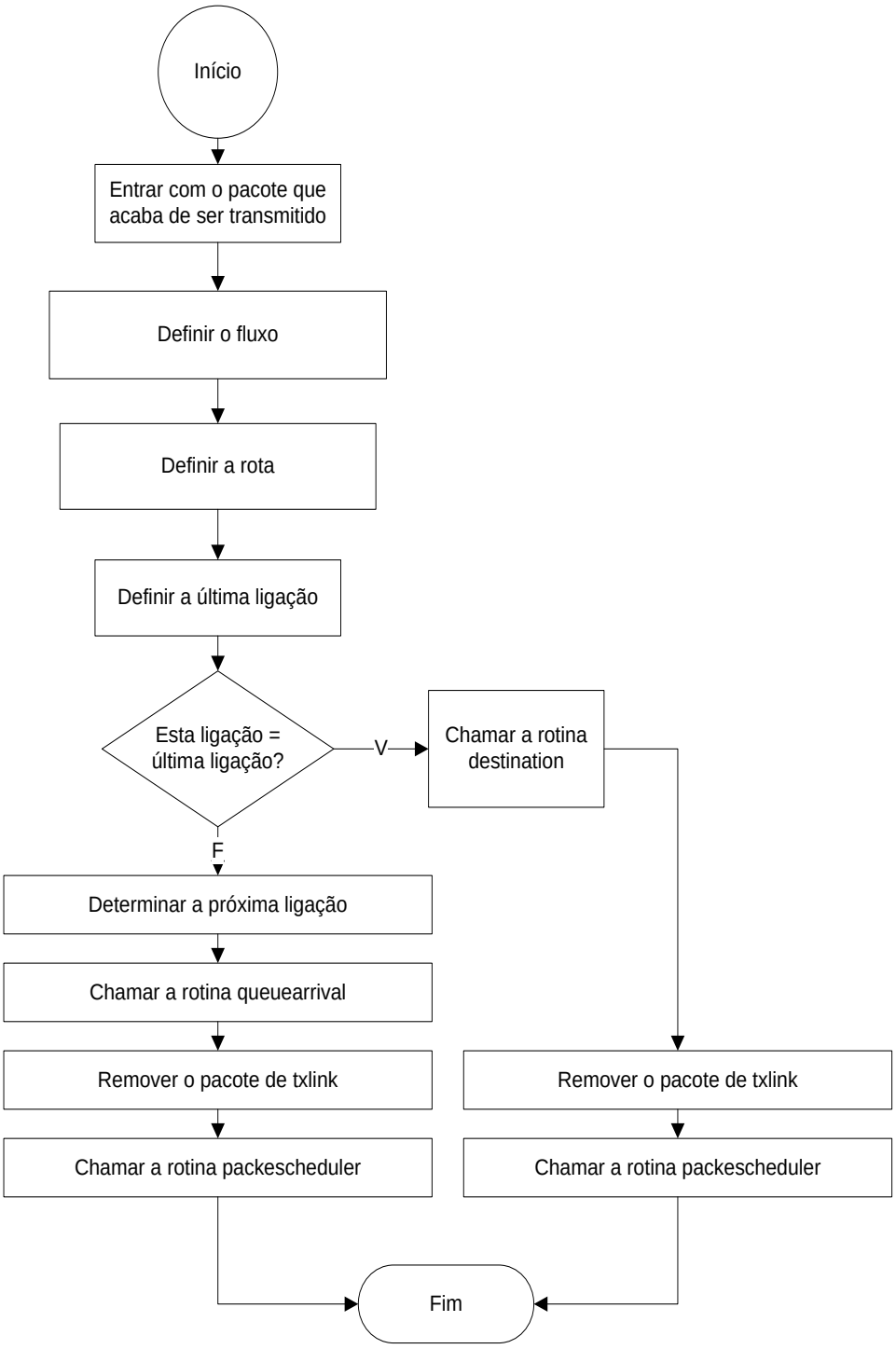
Observação:

- Esta rotina difere da poiexp apenas por gerar pacotes com comprimentos fixos, razão pela qual não se apresentam o fluxograma e a descrição da mesma.

5.8 Rotina txlinkdeparture

```
function txlinkdeparture(thisLink)

global TxLinks;
global Flows;

thisPacket=TxLinks{thisLink}; %Packet that ended
transmission
thisFlow=thisPacket(1); %Flow of this packet
thisFlowPath=Flows{thisFlow,2}; %Path of this flow
lastLink=thisFlowPath(end); %Last link in the flow path

if thisLink==lastLink %If this link is last link
    destination(thisPacket); %Send packet to destination
else
    nextLink=thisFlowPath(find(thisLink)+1); %Determines
next link in the flow path
    queuearrival(thisPacket,nextLink); %Send this packet to
next link
end

TxLinks{thisLink}=[]; %Removes this packet from this tx link

packetscheduler(thisLink); %Calls packet scheduler of this
link
```

5.8.1 Fluxograma da rotina txlinkdeparture



5.8.2 Descrição

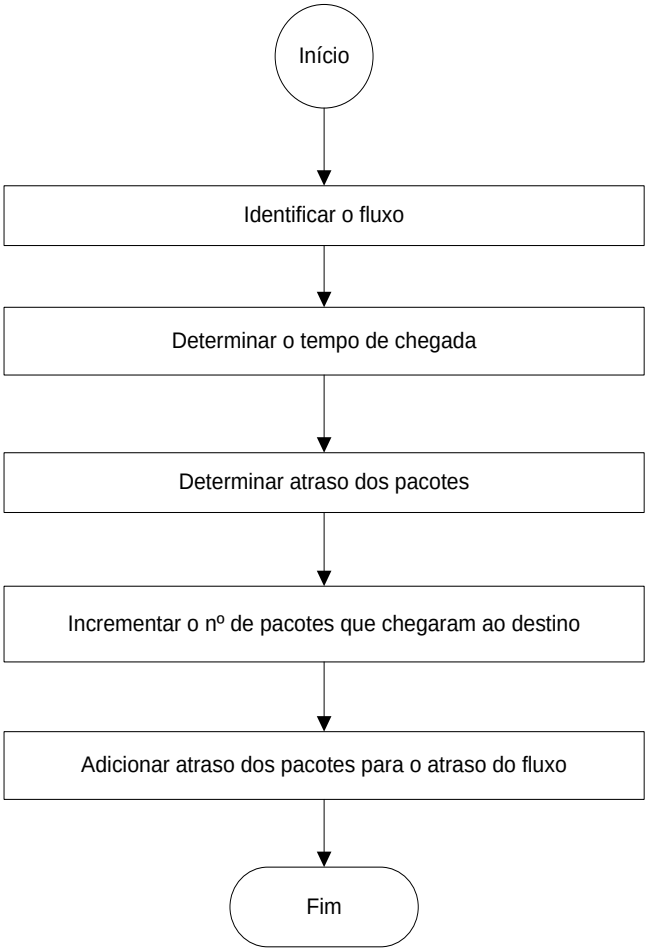
Quando esta rotina é chamada, definem-se o fluxo de pacote, o caminho do fluxo e a última ligação no caminho do fluxo. O pacote que acaba de ser transmitido é armazenado em TxLinks{thisLink}.

Processa partidas de pacotes em uma ligação de transmissão e determina o próximo recurso para o pacote que acaba de ser transmitido, que pode ser uma ligação ou o destino e encaminha-o para este recurso. Se a ligação for a última então o pacote é enviado para o destino, ou seja é chamada a rotina destination(thisPacket), se não, determina-se a próxima ligação no caminho do fluxo - nextLink=thisFlowPath(find(thisLink)+1) e é chamada a rotina queuearrival(thisPacket,nextLink) e o pacote é removido de TxLinks{thisLink} e então enviado para a próxima ligação de acordo com o algoritmo de escalonamento seleccionado pela função packetscheduler(thisLink.

5.9 Rotina destination

```
function destination(thisPacket)

global Time;
global FlowStats;

thisFlow=thisPacket(1); %Flow of this packet
thisArrivalTime=thisPacket(3); %Arrival time of this packet
thisPacketDelay=Time-thisArrivalTime; %Calculates packet
delay
FlowStats{thisFlow}(1)=FlowStats{thisFlow}(1)+1; %Increments
number of packets that arrived at destination
FlowStats{thisFlow}(2)=FlowStats{thisFlow}(2)+thisPacketDela
y; %Adds delay of this packet to flow delay
```

5.9.1 Fluxograma da rotina destination



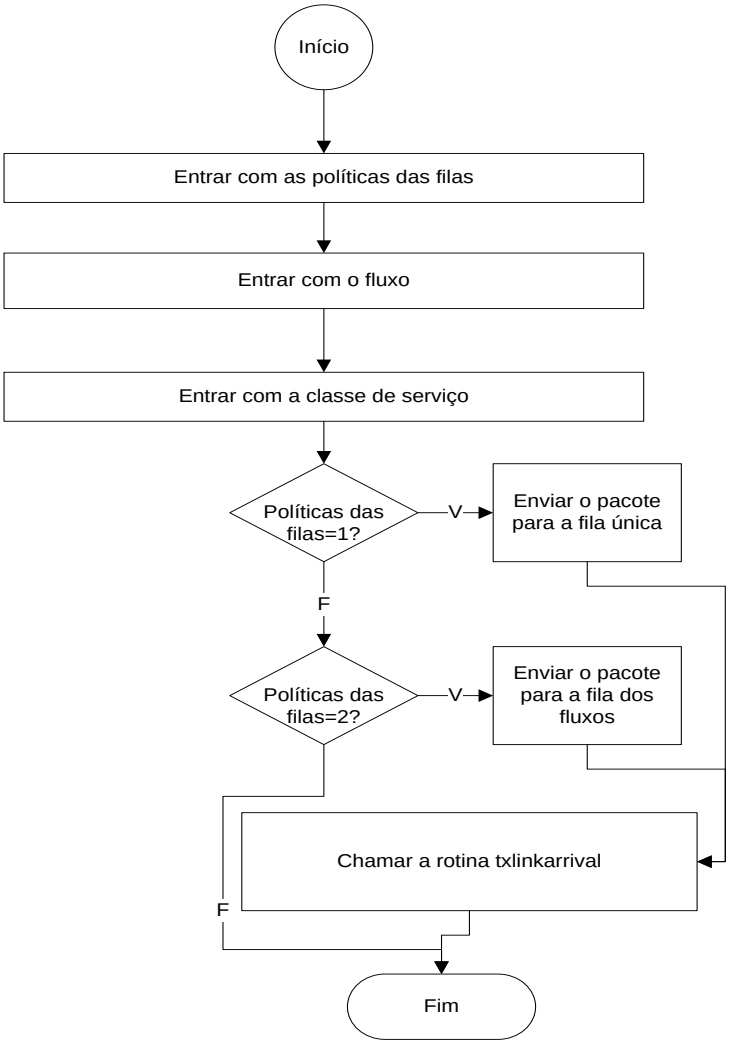
5.9.2 Descrição

Os pacotes transmitidos são encaminhados para esta rotina. A rotina define o fluxo de pacotes – thisFlow=thisPacket(1), o tempo de chegada dos pacotes - thisArrivalTime=thisPacket(3), calcula o atraso dos pacotes - thisPacketDelay=Time-thisArrivalTime, incrementa ou faz a contagem do número de pacotes que chegaram ao destino - FlowStats{thisFlow}(1)=FlowStats{thisFlow}(1)+1 e adiciona o atraso destes pacotes para o atraso do fluxo – FlowStats{thisFlow}(2) =FlowStats{thisFlow}(2)+thisPacketDelay.

5.10 Rotina queuearrival

```
function queuearrival(thisPacket,thisLink);

global Queues;
global LinkQueuingPolicy;

 %disp('in queuearrival');

thisLinkQueuingPolicy=LinkQueuingPolicy(thisLink);
thisFlow=thisPacket(1);

switch thisLinkQueuingPolicy
    case 1 %Queuing policy is single queue
        Queues{thisLink,1}(end+1,:)=thisPacket; %Enqueue
this packet at single queue
    case 2 %Queuing policy is per flow
        Queues{thisLink,thisFlow}(end+1,:)=thisPacket;
%Enqueue this packet at queue of this flow
end

txlinkarrival(thisLink); %Calls routine to deal with
(possible) arrival at tx link
```

5.10.1 Fluxograma da rotina queuearrival



5.10.2 Descrição

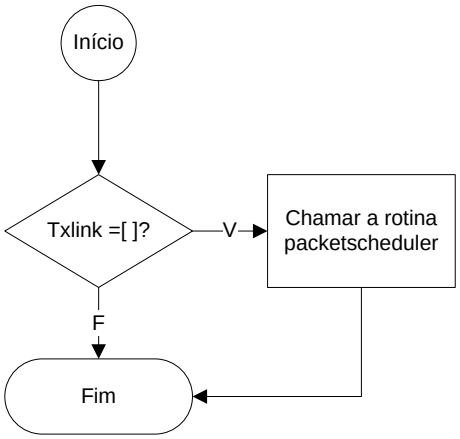
Esta rotina processa chegadas de pacotes a filas de uma ligação. Determina a fila para onde o pacote deve ser encaminhado de acordo com as políticas das ligações, e encaminha o pacote seleccionado para esta ligação caso a fila estiver vazia. Escolhe uma das políticas de ligação: 1 – o pacote é colocado na fila única: Queues{thisLink,1}(end+1,:)=thisPacket e

2 – O pacote é colocado na fila dos fluxos: Queues{thisLink,thisFlow}(end+1,:)=thisPacket. Em seguida chama a rotina txlinkarrival(thisLink).

5.11 Rotina txlinkarrival

```
function txlinkarrival(thisLink)

global TxLinks;

if isempty(TxLinks{thisLink}) %If tx link is empty
   packetscheduler(thisLink); %Call packet scheduler
end
```

5.11.1 Fluxograma da rotina txlinkarrival



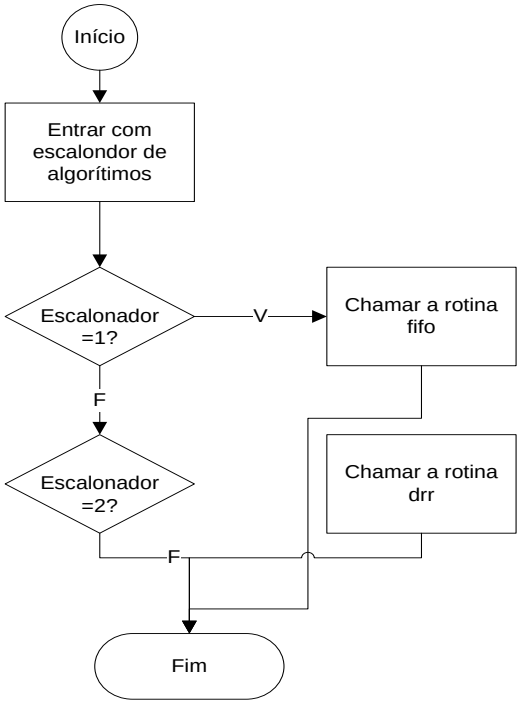
5.11.2 Descrição

Esta rotina processa chegadas de pacotes a uma ligação de transmissão. Se a ligação de transmissão estiver vazia, chama o pacote seleccionado para esta ligação. Caso contrário não faz nada.

5.12 Rotina packetscheduler

```
function packetscheduler(thisLink)

%This routine selects the link scheduling algorithm of this
link.

global LinkSchedulingAlgorithm;

thisLinkSchedulingAlgorithm=LinkSchedulingAlgorithm(thisLink
);

switch thisLinkSchedulingAlgorithm
    case 1
        fifo(thisLink); %Call fifo

    case 2
        drr(thisLink); %Call drr

end
```

5.12.1 Fluxograma da rotina packetscheduler



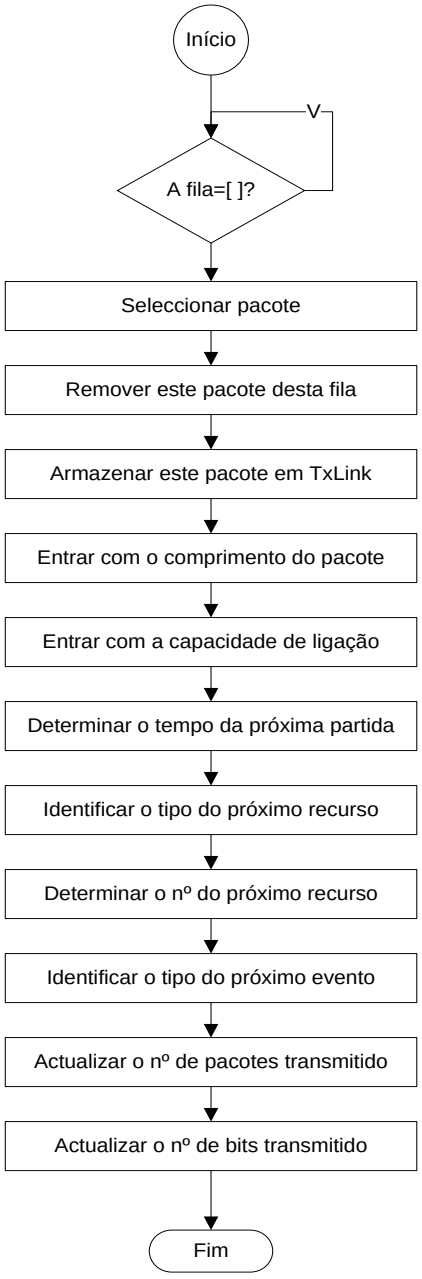
5.12.2 Descrição

A rotina packetscheduler(thisLink) selecciona o escalonador das ligações de acordo com as politicas de encaminhamento: 1 – chama o algoritmo **First – In – First – Out (FIFO)** e 2 – chama o algoritmo **Deficit Round Robin (DRR)**

5.13 Rotina fifo

```
function fifo(thisLink)

global Queues;
global TxLinks;
global LinkCapacity;
global EventList;
global Time;
global LinkStats;

if isempty(Queues{thisLink,1}) %If queue is empty
    return; %do nothing
end

%Selects packet
thisPacket=Queues{thisLink,1}(1,:); %Packet selected for
transmission

%Transfers this packet from queue to transmission link
Queues{thisLink,1}(1,:)=[]; %Removes selected packet from its
queue
TxLinks{thisLink}=thisPacket; %Stores this packet at this tx
link

%Schedules departure of this packet
thisLength=thisPacket(4); %Length of this packet
thisCapacity=LinkCapacity(thisLink); %Capacity of this link
nextDepartureTime=Time+thisLength/thisCapacity; %Time of next
departure
nextResourceType=2; %Next resource is link
nextResourceNumber=thisLink; %Next link is this link
nextEventType=2; %Next event type is departure
EventList(end+1,:)=[nextDepartureTime nextResourceType
nextResourceNumber nextEventType]; %Places departure event in
event list

%Updates de number of packets and the number of bytes that have
left this
%link
LinkStats{thisLink,1}(1)=LinkStats{thisLink,1}(1)+1;
LinkStats{thisLink,1}(2)=LinkStats{thisLink,1}(2)+thisLength;
```

5.13.1 Fluxograma da rotina fifo



5.13.2 Descrição

Neste algoritmo o escalonador é constituído por uma única fila e os pacotes são servidos pela ordem de chegada. Esta rotina determina o próximo pacote a ser transmitido, transfere este pacote desta fila para a ligação de transmissão e determina o tempo de partida. Após a partida de um pacote, uma fila vazia pode se formar.

No processamento, se a fila estiver vazia – isempty(Queues{thisLink,1}), o escalonador não faz nada, caso contrário selecciona o pacote a ser transmitido –

thisPacket=Queues{thisLink,1}(1,:). Como os pacotes são vectores formados por uma linha e várias colunas (no caso seis), este comando selecciona o primeiro elemento de todas as colunas. Em seguida transfere este pacote da fila para a ligação de transmissão, seno que para isso, remove o pacote da fila – Queues {thisLink,1}(1,:)=[ ] – e armazena-o em uma rotina chamada txlink – TxLinks{thisLink}=thisPacket. Em seguida, o escalonador prepara a partida do pacote. A realização desta acção exige a procura do comprimento do pacote seleccionado –thisLength=thisPacket(4), isto é, o comprimento dos pacotes corresponde ao elemento da quarta coluna do vector thisPacket determinada pela rotina poiexp ou poifix.

A capacidade da ligação é verificada através do comando thisCapacity=LinkCapacity(thisLink) determinada pela rotina parameters. Determinam-se também, o tempo da próxima partida, o tipo do próximo recurso (neste caso é uma ligação), o número do próximo recurso que é a ligação considerada, determina-se a natureza ou tipo do evento (partida). Esta rotina cria ainda uma lista de eventos, onde armazena-se as informações referentes às partidas.

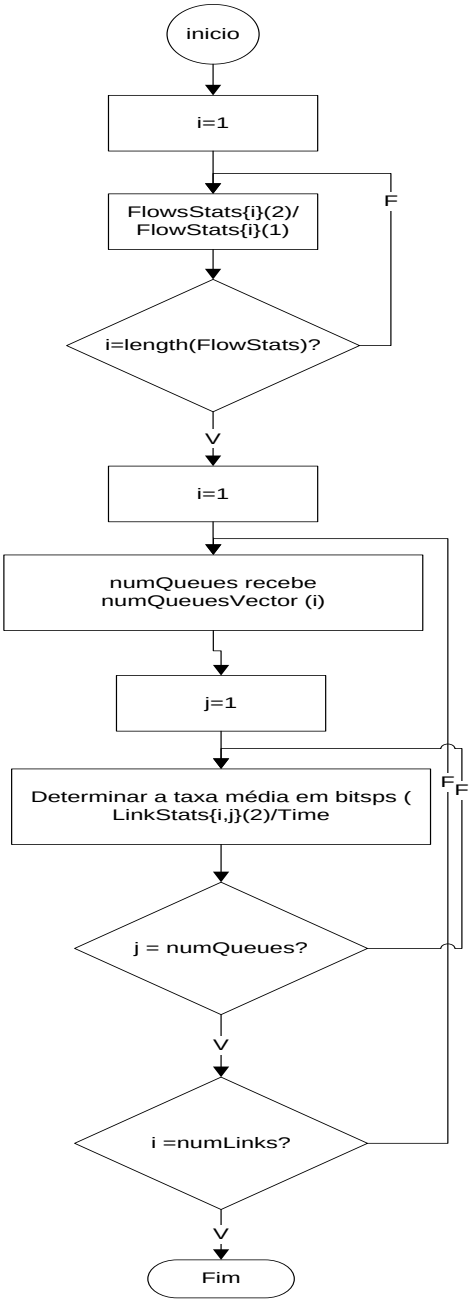
Finalmente, a rotina contabiliza ou actualiza o número de pacotes e o número de bits na ligação através dos comandos: LinkStats{thisLink,1}(1)=LinkStats{thisLink,1}(1)+1; LinkStats{thisLink,1}(2)=LinkStats{thisLink,1}(2)+thisLength.

5.14 Rotina report

```
function report
global FlowStats;
global LinkStats;
global numLinks;
global Time;
global numQueuesVector;
global numFlows;

 for i=1:length(FlowStats)
    disp('Average delay in flow');
    disp(i);
    disp(' = ');
    disp(FlowStats{i}(2)/FlowStats{i}(1));
end

for i=1:numLinks
    numQueues=numQueuesVector(i);
    for j=1:numQueues
        disp('Average rate (bits/sec) in queue');
        disp(j);
        disp('of link');
        disp(i);
        disp(' = ');
        disp(LinkStats{i,j}(2)/Time);
```

```
        end
    end
```

5.14.1 Fluxograma da rotina report



5.14.2 Descrição

Os resultados das simulações são requeridos por esta rotina.

5.15 Simulação dos exemplos apresentados no capítulo 4 (Aproximação de Kleinrock)

Para simular os exemplos apresentados no capítulo anterior, relativos a utilização da aproximação de Kleinrock, no simulador pnet, são feitas algumas alterações, ou seja, devem ser atribuídos na rotina parameters os valores adequados para cada parâmetro de entrada. Fundamentalmente deve-se fazer alterações nos parâmetros da rotina parameters.

O cenário 1 estabelece que todos os ramos devem ter a mesma capacidade 32kbits (ver os dados na secção 4.2).

A topologia a seguir apresentada determina que existe uma ligação que vai do ponto 1 para o ponto 2, outra que vai do ponto 2 para o ponto 4 e finalmente uma terceira ligação do ponto 4 para o ponto 3 conforme a figura 4.3 do capítulo anterior.

```
Topology=[0 1 0 0;
          0 0 0 2;
          0 0 0 0;
          0 0 3 0];
numLinks=3;
LinkCapacity=[32000 32000 32000]; %Link capacities, in bits/sec

LinkQueuingPolicy=[1 1 1]; %Link queuing policy can be 1 - single
queue.
LinkSchedulingAlgorithm=[1 1 1]; %Link scheduling algorithm can be 1 -
fifo.
Flows={[1,1/16.67,600],[1];[1,1/16.67,600],[2];[1,1/26.67,600],[3];[1,
1/10,400],[1 2]; [1,1/10,400],[3]}; %Flows is a cell array that stores
%for each flow the source type,the mean interarrival time (in
%seconds), the mean packet length (in bits)and the route.

endTime=1000*(1/16); %End time of simulation
```

De acordo com os parâmetros atribuídos nesta rotina, existem três ligações, cada uma com 32kbits da capacidade. Selecciona-se a política da fila única para todas as ligações e o algoritmo de encaminhamento FIFO. Todos os pacotes têm comprimentos exponencialmente distribuídos. Os fluxos x_1 representado por `{[1,1/16.67,600],[1]}`, x_2 representado por `{[1,1/16.67,600],[2]}`,e x_3 representado por `{[ 1,1/26.67,600],[3]}`, seguem as rotas [1], [3] e [2]

respectivamente, enquanto o $x_{4,}$ representado por `{[1,1/10,400],[1 2]; [1,1/10,400],[3]}` segue as rotas [1 2] e [3], (ver a figura 4.3).

O atraso médio que os pacotes dos fluxos levam para atravessar o percurso [1,2,4,3] será a igual soma a dos atrasos verificados em cada ramo.

Na Tabela 2.4 encontram-se os valores simulados para o cenário 1 conforme os parâmetros utilizados e com alguns ajustes, para o cenário 2,ou seja, se a taxa média do fluxo 3 passar de 10 Kbps para 16 Kbps um novo valor para o atraso médio será encontrado.

Como referido anteriormente pode-se comprovar que a aproximação de Kleinrock pode ser má, para ligações ponto-a-ponto em série, quando o tráfego é elevado.

Considerando o exemplo do capítulo anterior secção 4.2, e fazendo algumas alterações nos parâmetros de entrada, na rotina parameters encontra-se resultados que realmente comprovam este facto.

Levando em conta a figura 4.2, pode-se notar que para este caso existe um único fluxo de pacotes. Os pacote têm todos o mesmo tamanho e a mesma taxa de chegada e o fluxo percorre o trajecto ABC definido no programa de simulação como [1 2], conforme determinado pelo código topology.

```
Topology=[0 1 0;
          0 0 2;
          0 0 0];
numLinks=2;
LinkCapacity=[64000 64000]; %Link capacities, in bits/sec

LinkQueuingPolicy=[1 1]; %Link queuing policy can be 1 - single queue

LinkSchedulingAlgorithm=[1 1]; %Link scheduling algorithm can be 1 –
fifo.
Flows={[2,1/1000,16],[1 2]}; %Flows is a cell array that stores. For
each flow the source type,the mean interarrival time (in %seconds),
the mean packet length (in bits)and the route.

endTime=2; %End time of simulation
```

O resultado desta simulação encontra-se indicado na Tabela 2.5 do capítulo anterior.

CAPÍTULO VI – ALGORITMOS DE ESCALONAMENTO PARA DIFERENCIAÇÃO DA QUALIDADE DE SERVIÇO

Introdução

A diferenciação da qualidade de serviço é um requisito importante das futuras redes de comunicações de dados, nomeadamente da Internet. Para que essa diferenciação seja possível, a rede deve incluir um conjunto de mecanismos, incluindo algoritmos de escalonamento de pacotes.

Nos capítulos anteriores, fez-se referência aos algoritmos FIFO e Prioridade Estrita, todavia, propõe-se neste capítulo analisar os seus funcionamentos, bem como o do algoritmo Deficit – Round – Robin e estabelecer a comparação em termos das vantagens e inconvenientes existentes entre os mesmos.

6.1 Algoritmos de Escalonamento

Os algoritmos de escalonamento decidem a ordem pela qual, num servidor de pacotes (por exemplo, na ligação de saída de um comutador), os pacotes pertencentes a diferentes fluxos são servidos.

Os algoritmos podem ser classificados relativamente ao tipo de ordenação utilizado, da seguinte forma: por ordem de chegada (FIFO), de uma forma rígida (prioridade estrita), de uma forma rotativa (Round Robin, Weighted Round Robin, Deficit Round Robin), e por distribuição ponderada da largura de banda (Weighted Fair Queuing, Self Clock Fair Queuing).

6.1.1 First – In – First – Out (FIFO)

Neste algoritmo os pacotes são simplesmente servidos por ordem de chegada. Este sistema não requer qualquer processamento de ordenação. Por outro lado, ao tratar todo o tráfego de igual modo não permite diferenciar a qualidade de serviço oferecida, assim como não garante a existência de limites máximos nos atrasos. Além disso, o envio por ordem de chegada permite que fluxos que gerem n vezes mais tráfego recebam n vezes mais serviço.

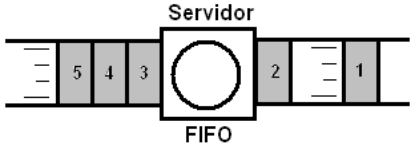


Figura 6.1- Ilustração do algoritmo FIFO

6.1.2 Prioridade Estrita

Neste algoritmo, o escalonador é constituído por várias filas de espera, cada uma com uma prioridade diferente. Os fluxos são classificados em diferentes níveis de prioridade e associados a uma determinada fila de espera de acordo com a sua prioridade.

O sistema de prioridade estrita assegura que o tráfego classificado como de maior prioridade é sempre servido antes do tráfego classificado como de menor prioridade. O sistema não requer qualquer processamento de ordenação; no entanto, requer que o tráfego seja classificado de acordo com a prioridade. Ao contrário do FIFO permite diferenciação da qualidade de serviço. No entanto, a discriminação entre o serviço dado às diferentes prioridades pode ser exagerada: se não existir nenhum mecanismo de controlo de admissão dos fluxos com maior prioridade, uma grande quantidade de pacotes de elevada prioridade pode impedir completamente o serviço de pacotes com menor prioridade (este fenómeno é usualmente denominado de starvation). Portanto, recomenda-se a utilização deste mecanismo de prioridade para tráfego que exija garantias muito estritas de diferenciação de qualidade de serviço.

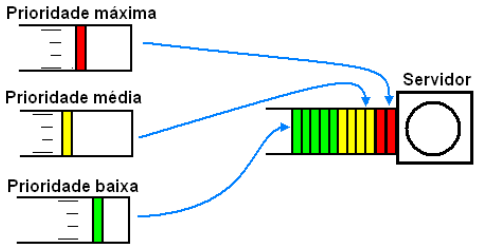


Figura 6.2 - Ilustração do algoritmo de prioridade estrita.

6.1.3 Deficit – Round – Robin (DRR)

O algoritmo Deficit – Round – Robin pretende distribuir uniformemente a largura de banda disponível utilizando um processo independente do comprimento dos pacotes. Este mecanismo tenta, em cada ciclo, servir uma quantidade de tráfego em octetos pré-definida e fixa que se designa por limiar.

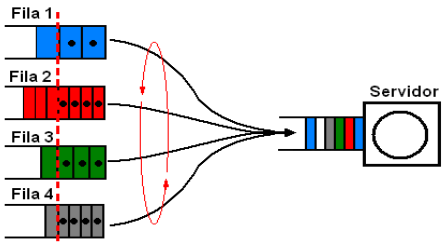


Figura 6.3 - Ilustração do funcionamento do mecanismo Deficit Round Robin.

Nesta figura, os pacotes assinalados com um ponto são os escolhidos para serem servidos, por se encontrarem a jusante do limiar.

O algoritmo Deficit Round Robin associa a cada fila i um crédito c_i, cujo valor inicial é zero e um limiar L_i. O limiar é uma constante característica da fila; o crédito pode variar de ciclo para ciclo e só pode tomar valores não-negativos. Em cada visita à fila i ao crédito disponível é incrementado um valor igual ao limiar, ou seja, $c_i = c_i + L_i$. Nessa visita, o sistema pode servir um ou mais pacotes até esgotar o crédito disponível. Concretamente, após ser servido um pacote de comprimento T octetos, o crédito é actualizado para c_i = max (0, c_i – T). Existindo um outro pacote na mesma fila, este pacote pode também ser servido se o seu comprimento for menor ou igual que o crédito disponível; caso contrário o algoritmo passa à próxima fila. Como é óbvio, um pacote só poderá ser servido num dado ciclo se houver crédito suficiente para servir todos os

seus octetos. Quando uma fila fica vazia o crédito respectivo é colocado a zero; caso contrário, uma fila poderia estar a acumular créditos indefinidamente conduzindo eventualmente a condições de injustiça.

6.1.3.1 Exemplo da aplicação do escalonador drr

Considere um sistema com três filas de espera, em que nas filas 1, 2 e 3 estão pacotes com comprimentos 1500, 800 e 1200 octetos, respectivamente, à espera de ser servidos. O limiar é 1000 e o crédito inicial é 0 em todas as filas de espera. No primeiro ciclo o sistema não serve a fila 1 pois o crédito disponível C_1 = 1000 é inferior ao comprimento do pacote; serve a fila 2, ficando com um crédito de C_2 = 1000-800 = 200; não serve a fila 3 pois o crédito disponível C_3 é inferior ao comprimento do pacote. No segundo ciclo, o crédito da fila 1 é incrementado para C_1 = 2000. Agora o pacote já pode ser servido e sobra um crédito de C_1 = 2000-1500 = 500. O crédito da fila 3 também é incrementado para C_3 = 2000. O pacote é servido sobrando um crédito de C_3 = 2000 – 1200 = 800. O crédito na fila 2 vai a zero pois esta fila não tem pacotes para transmitir.

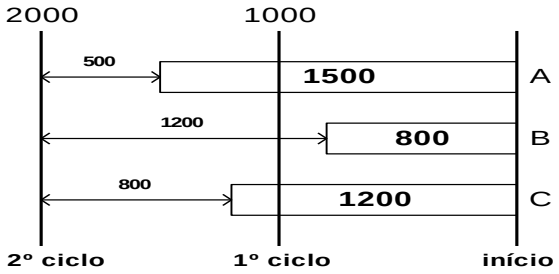


Figura 6.4 - Exemplo de operação do algoritmo Deficit – Round – Robin

A distribuição de largura de banda de uma forma não equitativa, pode ser conseguida atribuindo limiares diferentes a cada fila de espera. De notar que, preferencialmente, não deve verificar-se uma situação em que o valor de todos os limiares seja inferior ao comprimento máximo dos pacotes. Neste caso, surgiriam situações em que poderia não ser servido qualquer pacote num ciclo, obrigando a uma actualização desnecessária dos valores dos créditos e aumentando portanto a complexidade associada aos cálculos.

Por ser um sistema rotativo, não consegue evitar que o tráfego de cada fluxo seja acumulado e servido todo de uma vez (até ao limiar), o que aumenta as variações do

atraso. Este problema é mais acentuado em fluxos que utilizem uma elevada percentagem da largura de banda (limiares elevados) e que tenham pacotes pequenos.

6.2 Comparação entre os algoritmos de escalonamento

Tabela 2.6 – Comparação entre os algoritmos de escalonamento

Designação	Sumário	Vantagens	Desvantagens
First-come-first-served (FIFO)	• Pacotes são servidos pela ordem de chegada	• Facil de implementar; • O sistema não requer qualquer processamento de ordenação.	• Não permite diferenciar a qualidade do serviço oferecido; • Não garante a existência de limites máximos nos atrasos.
Prioritização estrita (PQ)	• Cada fila tem uma prioridade e o tráfego é classificado de acordo com a prioridade.	• O sistema não requer qualquer processamento de ordenação; • Permite diferenciação da qualidade do serviço.	• Pode ocorrer o fenómeno denominado de starvation.
Deficit round robin (DRR)	• Distribui uniformemente a largura de banda disponível usando um processo independente do comprimento dos pacotes.	• Serve em cada ciclo uma quantidade de tráfego em octetos pré-definida e fixa que se designa por limiar; • Associa a cada fila um crédito.	• Um pacote só poderá ser servido num dado ciclo se houver crédito suficiente para servir todos os seus octetos.

6.3 Simulação de redes com diferenciação de Qualidade de Serviço

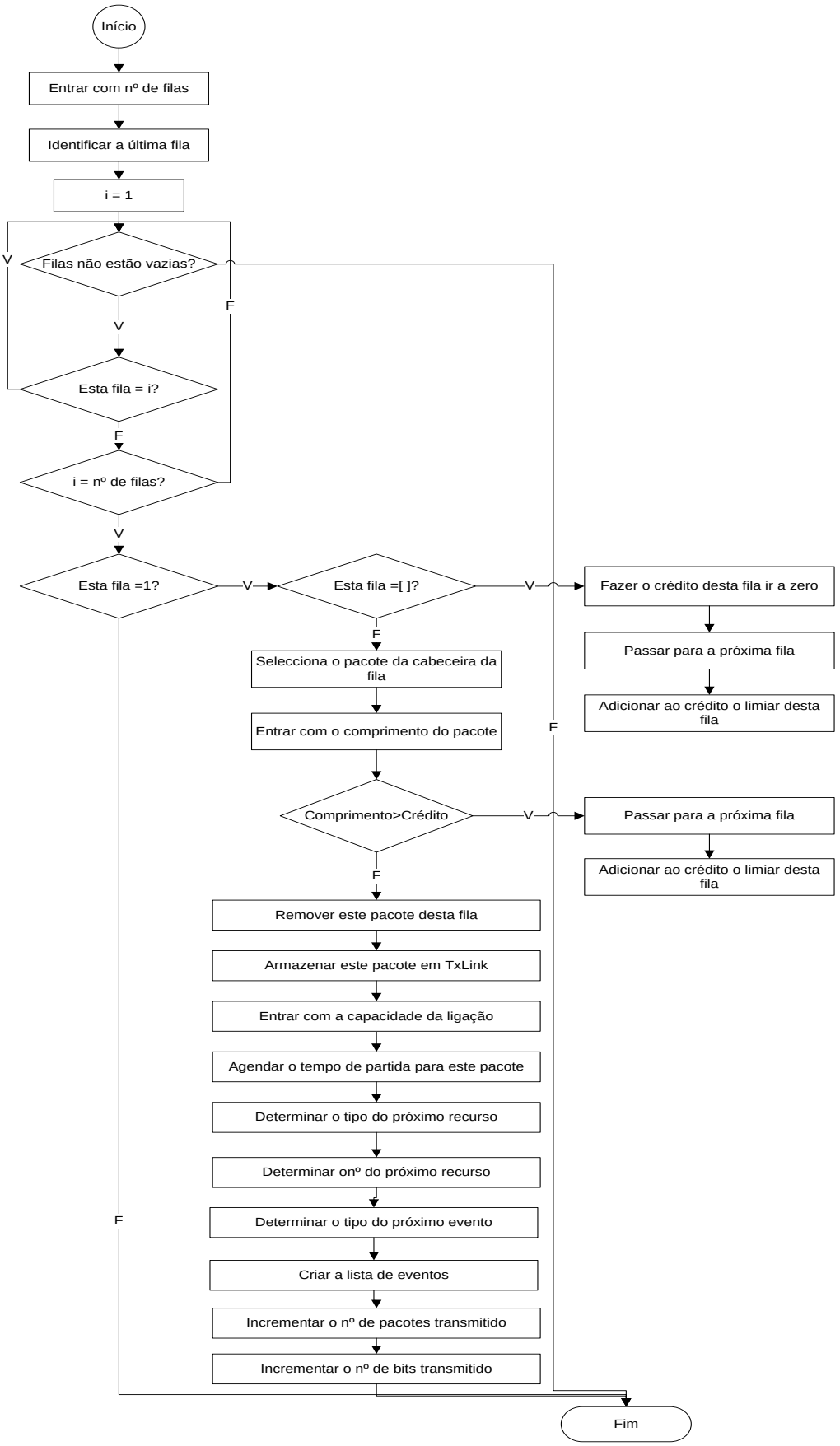
Para além dos escalonadores Prioridade Estrita e Fifo analisadas nos capítulos 3 e 5, propõe-se ainda, neste capítulo, a implementação do escalonador Deficit – Round – Robin (DRR).

6.3.1 Rotina drr

```
function drr(thisLink)

global numQueuesVector;
global Queues;
global thisPacket;
global TxLinks;
global LinkCapacity;
global EventList;
global Time;
global LinkStats;
global lastQueue;
global thisLength;
global DRRThreshold;
global DRRCredit;


thisQueue=lastQueue(thisLink);
numQueues=numQueuesVector; %Number of queues at link
for i=1:numQueues
    if not(isempty(Queues{thisLink,i}))
        break;

    elseif i==thisQueue
        return
    end
end

while 1
    if isempty(Queues{thisLink,thisQueue}) %If this queue is
empty
        DRRCredit(thisLink,thisQueue)=0; %Reset credit of
this queue to zero
        thisQueue=mod(thisQueue,numQueues)+1; %Move to next
queue

DRRCredit(thisLink,thisQueue)=DRRCredit(thisLink,thisQueue)+D
RRThreshold(thisLink,thisQueue); %Add threshold to credit
    else
        thisPacket=Queues{thisLink,thisQueue}(1,:); %Read
packet in the head of queue
        thisLength=thisPacket(4); %Length of this packet
        if thisLength>DRRCredit(thisLink,thisQueue)
            thisQueue=mod(thisQueue,numQueues)+1; %Move to
next queue

DRRCredit(thisLink,thisQueue)=DRRCredit(thisLink,thisQueue)+D
RRThreshold(thisLink,thisQueue); %Add threshold to credit
```

```
        else
            %Transfers this packet from queue to transmission
link and updates credit
            Queues{thisLink,thisQueue}(1,:)=[]; %Removes
selected packet from its queue
            TxLinks{thisLink}=thisPacket; %Stores this packet
at this tx link

LinkStats{thisLink,thisQueue}(1)=LinkStats{thisLink,thisQueue
}(1)+1; %Updates number of packets that left the queue

LinkStats{thisLink,thisQueue}(2)=LinkStats{thisLink,thisQueue
}(2)+thisLength; %Updates number of bytes that left the queue

DRRCredit(thisLink,thisQueue)=DRRCredit(thisLink,thisQueue)-
thisLength;
            %Schedules departure of this packet
            thisCapacity=LinkCapacity(thisLink); %Capacity of
this link
            nextDepartureTime=Time+thisLength/thisCapacity;
%Time of next departure
             nextResourceType=2; %Next resource is link
             nextResourceNumber=thisLink; %Next link is this
 link
             nextEventType=2; %Next event type is departure
             EventList(end+1,:)=[nextDepartureTime
            nextResourceType  nextResourceNumber
            nextEventType];%Places departure event in event
            list
             lastQueue(thisLink)=thisQueue;
            break;
        end
    end
end
```

6.3.1.1 Fluxograma da rotina drr



6.3.1.2 Descrição

O algoritmo Deficit – Round – Robin pretende distribuir uniformemente a largura de banda disponível utilizando um processo independente do comprimento dos pacotes
É uma rotina onde se consideram várias filas de espera e portanto, define-se o número de filas através do comando numQueues=numQueuesVector e identifica-se a última fila visitada - thisQueue=lastQueue(thisLink), sendo que neste caso a última fila visitada é a fila 1 conforme determinada na rotina init.
Quando a rotina é chamada verifica-se primeiro se existe mais algum pacote nesta fila - isempty(Queues{thisLink,thisQueue})e, existindo, selecciona-se o pacote a transmitir - thisPacket=Queues{thisLink,thisQueue}(1,:), analisa-se o seu comprimento e compara-o com o crédito disponível. Caso o comprimento seja maior do que o crédito disponível, passa-se para a próxima fila - thisQueue=mod(thisQueue,numQueues)+1 – havendo pacotes nesta fila então soma-se ao crédito da fila o limiar desta fila- DRRCredit(thisLink,thisQueue)=DRRCredit(thisLink,thisQueue)+DRRThreshold(thisLink,thisQueue), e verifica-se se pode ser transmitido algum pacote. Caso negativo passa-se à próxima fila. Se puder então a rotina transfere este pacote da fila para a ligação de transmissão e actualiza o crédito, ou seja, remove este pacote desta fila e o armazena em TxLinks, programa a partida do pacote, actualiza o número de pacotes e o número de bits através da fila, e assim sucessivamente. Se por acaso nenhuma das filas tem pacotes, então o escalonador não deve fazer nada.

6.3.2 Exemplo da aplicação do algoritmo Deficit-Round-Robin

Como anteriomente dito, o algoritmo Deficit – Round – Robin pretende distribuir uniformemente a largura de banda disponível utilizando um processo independente do comprimento dos pacotes.
Para comprovar este facto, propõe-se analisar o exemplo que se segue. Neste exemplo, considera-se uma ligação entre dois pontos com 64Kbps de capacidade. No primeiro caso considera-se que os dois fluxos de pacotes tenham comprimentos diferentes e na segunda situação que os pacotes tenham comprimentos iguais. O limiar de uma das filas é duas vezes maior do que da outra.

Com estas condições e outras apresentadas na rotina parameters abaixo, conclui-se que através do algoritmo Deficit – Round – Robin, pode-se distribuir uniformemente a largura de banda disponível utilizando um processo independente do comprimento dos pacotes, pois o número de Bits/segundo transmitidos do fluxo pertencente à fila cujo limiar e duas vezes maior e também em média duas vezes maior tanto para o caso em que os pacotes tenha mtodos o mesmo tamanho como para o caso de terem tamanhos diferentes.

Rotina parameters:

```
Topology=[0 1;
          0 0];
%Links
numLinks=1;
LinkCapacity=[64000]; %Link capacities, in bits/sec
LinkQueuingPolicy=[2]; %Link queuing policy can be 1 - single queue or
%2 - per flow
LinkSchedulingAlgorithm=[2]; %Link scheduling algorithm can be 1 -
%fifo or 2 - deficit round-robin.
Flows={[2,1/48,1000],[1];[2,1/48,2000],[1]}; %Flows is a cell array
%that stores for each flow the source tipy, the mean interarrival time
%(in seconds), the mean packet length (in bits)and the route.
% %Scheduler parameters
DRRThreshold=[2000,1000]; %Matrix with the threshold of each queue in
%each link; links are rows; queues are columns; no queue in a given
%link means zero threshold
DRRCredit=[0,0]; %Matrix with the credit of each queue in each link;
%links are rows; queues are columns; no queue in a given link means
%Inf credit
% %End time of simulation
endTime=1000*(1/48);
```

A Tabela 6.1 mostra os resultados encontrados através da simulação em MATLAB usando o algoritmo DRR.

Tabela 6. 1- Resultados do exemplo do escalonador DRR

Fluxo	Nº ligações	Capacidade da ligação	Limiar	Fila	Bits/segundo transmitidos	Relação de bits/s transmitidos	Comprimentodos pacotes
1	1	64kbis/s	2000	1	$4{,}2189\times10^4$	bits/s do fluxo 1 → 2×btis/s do fluxo 2	Diferentes
2			1000	2	$2{,}1022\times10^4$		
1	1	64kbis/s	2000	1	$4{,}2517\times10^4$	bits/s do fluxo 1 → 2×btis/s do fluxo 2	Iguais
2			1000	2	2.1259×10^4		

CONCLUSÃO

Diante do exposto ao longo de todo este trabalho, pode-se tirar as seguintes conclusões:

- A Internet é baseada em comutação de pacotes. Neste tipo de redes de comunicações o desenvolvimento de modelos para análise de desempenho requer o conhecimento da estrutura da rede e do tráfego oferecido;

- O estudo analítico das redes é normalmente baseado em modelos desenvolvidos a partir da teoria de filas de espera;

- O modelo de uma ligação ponto-a-ponto é um processo de nascimento e morte, mas apenas no caso de tempo de serviço exponencialmente distribuído;

- Uma ligação ponto-a-ponto com capacidade μ pacotes /*s* onde chegam pacotes a uma taxa de Poisson λ pacotes/*s* é um sistema M/G/1. Se o comprimento dos pacotes for exponencialmente distribuído degenera num sistema M/M/1. Se o comprimento dos pacotes for fixo degenera num sistema M/D/1;

- A partir dos dados apresentados nas Tabelas 2.1 e 2.3 pode-se concluir que em geral, quando o comprimento dos pacotes for fixo, os valores correspondentes a: número médio de pacotes no sistema, atraso médio no sistema, atraso médio na fila de espera, e ao número médio de pacotes na fila de espera são menores do que quanto o tamanho dos pacotes tem uma distribuição exponencial. No entanto, o factor de utilização do servidor permanece constante;

- Normalmente quando uma ligação é partilhada por fluxos de pacotes de dois ou mais utilizadores, há possibilidade de se utilizar um modelo que permita a atribuição de prioridades aos fluxos de acordo com a conveniência do sistema.

Neste contexto, alguns dos fluxos terão um atraso no sistema inferior aquele que teriam os fluxos de ambos os utilizadores se fossem misturados num mesmo buffer e servidos de acordo com uma disciplina FIFO;

- A comunicação entre dois terminais pode ser estabelecida por encaminhamento fixo ou dinâmico;

- A aproximação de Kleinrock, é um método de muita valia na avaliação do desempenho das redes de comunicações com encaminhamento fixo, mas é uma aproximação e como tal, tem limitações, principalmente quando se usam ligações em cascata, para um sistema cujo tráfego é elevado (ver a Tabela 2.5);

- A diferenciação da qualidade de serviço é um requisito importante das futuras redes de comunicações de dados, nomeadamente da Internet;

- A diferenciação da qualidade de serviço só é possível se a rede tiver um conjunto de mecanismos, incluindo algoritmos de escalonamento de pacotes;

- Existem vantagens e inconveniente entre os vários algoritmos de escalonamento, e o uso de um ou outro mecanismo depende da conveniência do sistema e do desempenho que se pretenda conseguir. O escalonador FIFO é um dos mais elementares, mas muito utilizado em redes de comunicações;

- O algoritmo Deficit – Round – Robin, permite distribuir uniformemente a largura de banda disponível utilizando um processo independente do comprimento dos pacotes;

- Para além dos mecanismos referenciados ao longo deste trabalho, existem outros de muita importância para redes com diferenciação da qualidade de serviço, tais como: Round-Robin (RR), Weighted Round-Robin (WRR), Generalized Processor Sharing (GPS), Weighted Fair Queuing (WFQ), assuntos de interesse que poderão ser objectos de futuras investigações.

REFERÊNCIAS BIBLIOGRÁFICAS

BARAN, P. *On Distributed Communications*. RAND Memorandum, 1964.

BERTSEKAS, D.; GALLAGER, R. *Data Networks*, 2nd edition; Prentice-Hall, 1992

CERF, V.G., e KAHN, R.E. *A Protocol for Packet Network Interconnection*. IEEE.Trans. Commun. COM-22, 5, 637-641, May 1974.

FONTANELLA, G. C; MORABITO, R. *Modelagem Por Meio De Teoria Defilas Do Tradeoff Entre Investir Em Canais De Atendimento E Satisfazer O Nível De Serviço Em Provedores Internet*. Universidade Federal de São Carlos Departamento de Engenharia de Produção 13565-905 - São Carlos – SP. **(s/d).** Disponível em: http://www.dep.ufscar.br/docentes/morabito/g&p97c.pdf. Consultado em 10 -02- 2010.

GOMES, T.; SIMÕES, C. *Algorítmos de Encaminhamento Dinânico e Atribuição do comprimento de Onda em redes WDM*. Disponível em: http://piano.dsi.uminho.pt/museu/CPI80/comum20.pdf. (s/d). Consultado em 10/02/2010.

HARDY, HENRY. *The History of the Net* [em linha]. Master's Thesis, School of Communications, Grand Valley State University [citado em 14 de Março de 1999].

KANHERE, S. S.; SETHU, H. (2001). *Fair, Efficient and Low-Latency Packet Scheduling Using Nested Deficit Round Robin*. Department of ECE Drexel University 3141 Chestnut Street, Philadelphia, PA 19104-2875. 2001. Disponível em:

http://cial.csie.ncku.edu.tw/2006pdf/Fair,%20efficient%20and%20low-latency%20packet%20scheduling%20using%20nested%20deficit%20round%20robin.pdf. Consultado em 15 de Junho de 2009.

KESHAV, S.; WESLEY, A. *An Engineering Approach to Computer Networking.* Projeto Final de Graduação – Universidade de Brasília, Faculdade de Tecnologia, Departamento de Engenharia Elétrica, 1977.

KLEINROCK, L. *Information Flow in Large Communications Nets.* RLE Quarterly Progress Report, Julho 1961.

KLEINROCK, L. *Communication Nets: Stochastic Message Flow and Delay*. McGraw Hill, New York, 1964.

SANTOS, H.J.S. *Análise e Interpretação da Dinâmica do Crescimento de Internet. Trabalho de Síntese*. Provas de Aptidão Pedagógica e Capacitação Científica. Universidade de Beira do Interiro. Portugal, 2001.

SHELDON, M. R. *Introdution to Probability Models*, 6th edition, Academic Press, 1997.

SHREEDHAR, M. e VARGHESE, G. *Efficient Fair Queuing Using Deficit-Round-Robin*. IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 4, NO. 3, JUNE 1996. Disponível em: http://pages.cs.wisc.edu/~akella/CS740/S08/740-Papers/DRR.pdf. Consultado em 15 de Junho de 2009.

SILVA, PAULO ROGÉRIO TARCHETTI. *Estudo do tráfego auto-similar em redes multiserviço* [Distrito Federal]. xi, 72p., 297mm (ENE/FT/UnB, Engenheiro, Engenharia de Redes de comunicação, 2005).

VAZ, Francisco. *Probabilidades e Processos Estocásticos para Engenharia Electrotécnica*. Universidade de Aveiro. Campus Universitário de Santiago 3810-193, 2002.